People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University of 8 May 1945-Guelma-
Faculty of Mathematics, Computer Science and Science of Matter
Department of Computer Science

Master Thesis

Specialty: Computer Science

Option:

Science and technology of information and communication

Theme

Binarization of Degraded Document Images Using Deep Learning Techniques

Presented by: Assala BEN KAMOUCHE

Jury Members:

N	Full name	Quality
1	Aicha AGGOUNE	Chairman
2	Abderrahmane KEFALI	Supervisor
3	Adil BOUGHIDA	Examiner

June 2025.

ملخص

التحويل الثنائي هو خطوة أساسية في مرحلة ما قبل المعالجة في مجال معالجة الصور وتحليلها، لا سيما في سياق صور الوثائق التاريخية المتدهورة. فمع مرور الزمن، تتعرض هذه الوثائق لأنواع متعددة من التدهور نتيجة سوء ظروف التخزين والعوامل البيئية، مما يجعل التحويل الثنائي الدقيق أمرًا ضروريًا للمهام اللاحقة مثل التحسين، والتعرف، والأرشفة.

تعتمد تقنيات التحويل الثنائي التقليدية، المعروفة عمومًا باسم طرق العتبة (thresholding)، على اختيار عتبة واحدة أو أكثر – سواء كانت عالمية أو محلية – للفصل بين النص (المقدمة) والخلفية. وعلى الرغم من أن هذه الأساليب دُرست على نطاق واسع، فإنها غالبًا ما تفشل في التعامل مع حالات التدهور المعقدة مثل تسرب الحبر، أو الإضاءة غير المتجانسة، أو الخلفيات المملوءة بالأنسجة.

لمعالجة هذه القيود، يقترح هذا المشروع مقاربة للتحويل الثنائي تعتمد على التعلم العميق، دون الاعتماد على تقنيات العتبة. وبشكل خاص، نعتمد على الشبكات العصبية الالتفافية (CNN) لتصنيف البكسلات، مستفيدين من قدرتها على تعلم السمات التمييزية مباشرةً من البيانات. وقد أثبتت الشبكات العصبية الالتفافية فعاليتها في التقاط المعلومات البنيوية والسياقية، مما يجعلها مناسبة جدًا للتحديات التي تفرضها الوثائق المتدهورة.

تم تدريب النظام المقترح وتقييمه باستخدام مجموعة متنوعة من الوثائق التاريخية المشروحة. وقد أظهرت النتائج التجريبية أن النهج المقترح يحقق أداءً تنافسيًا، بل ويتفوق غالبًا على الطرق التقليدية والحديثة، وفقًا لمقاييس التقييم المستخدمة في مسابقات التقييم العالمية. وتؤكد هذه النتائج على قوة النظام المقترح وقدرته على التعميم، مما يجعله مرشحًا واعدًا للتطبيقات الواقعية في تحليل وأرشفة الوثائق.

الكلمات المفتاحية *:* التحويل الثنائي، الوثائق المتدهورة، تحليل الوثائق والتعرف عليها، التعلم العميق، الشبكات العصبية الالتفافية، تحضير البيانات.

Abstract

Binarization is a fundamental preprocessing step in image processing and analysis, particularly in the context of degraded historical document images. Over time, such documents often suffer from various forms of deterioration due to poor storage conditions and environmental factors, making reliable binarization essential for subsequent tasks like enhancement, recognition, and archiving.

Traditional binarization techniques, widely referred to as thresholding methods, operate by selecting one or more global or local thresholds to separate text (foreground) from background. While these approaches have been extensively studied, they often fall short when dealing with complex degradations such as ink bleed-through, uneven illumination, or textured backgrounds.

To address these limitations, this project proposes a deep learning-based binarization approach that eliminates the reliance on thresholding. Specifically, we employ Convolutional Neural Networks (CNNs) for pixel-level classification, leveraging their ability to learn discriminative features directly from data. CNNs have proven effective in capturing structural and contextual information, making them particularly suitable for the challenges posed by degraded document images.

The proposed system was trained and evaluated on a diverse set of annotated historical documents. Experimental results demonstrate that our approach achieves competitive and often superior performance compared to classical and recent methods, as measured by evaluation metrics commonly used in international benchmarking competitions. These outcomes confirm the robustness and generalization ability of the proposed method, establishing its potential for real-world applications in document image analysis and digitization.

Keywords: *Binarization, degraded document, document analysis and recognition, deep learning, CNN, data oreparation.*

Acknowledgments

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

First and foremost, I thank and praise Allah Almighty for granting me the strength, patience, and perseverance to complete this research work.

I had the great honor of being supervised by Dr. *Abderrahmane Kefali*, a distinguished scholar and a true role model in the academic field. I am deeply grateful for his invaluable guidance, encouragement, and support throughout this journey.

I would also like to extend my sincere thanks to the members of the defense committee for their insightful feedback and constructive suggestions, which have greatly enriched the quality of this work.

My heartfelt appreciation goes to the faculty and staff of **the Computer Science Department** at the **University of 8 May 1945**. Their support and dedication provided an inspiring environment for academic growth and achievement.

Last but not least, I express my profound gratitude to my family and my dear friend, whose unwavering support and encouragement played a vital role in the realization of this project.

Table of Contents

List of Figures

List of Tables

General Introduction

The digitization of historical documents plays a pivotal role in the preservation, dissemination, and analysis of cultural and intellectual heritage. Libraries, museums, and research institutions around the world are continuously engaged in the process of converting fragile and often severely degraded physical documents into digital formats to ensure their long-term accessibility. However, this digitization process is frequently challenged by the poor visual quality of the source materials, which may exhibit various types of degradations such as faded ink, bleed-through, stains, uneven lighting, background textures, and physical damage to the paper medium. These degradations pose a serious barrier to both human readability and the effectiveness of automated document analysis systems.

A fundamental preprocessing task that underlies most document image analysis applications—such as Optical Character Recognition (OCR), layout analysis, keyword spotting, and semantic segmentation—is binarization. This operation involves separating the text (foreground) from the background by converting a grayscale or color image into a binary image. While it appears conceptually simple, binarization is, in practice, a complex and ill-posed problem, especially when dealing with historical documents characterized by irregular and non-uniform degradations.

From a computational point of view, binarization can be seen as a classification task at the pixel level, where each pixel must be labeled as either foreground or background based on its visual characteristics and context. Classical binarization methods, such as global thresholding (e.g., Otsu's method) and local thresholding (e.g., Sauvola, Niblack), rely heavily on heuristic rules and intensity statistics, and although they are computationally efficient, they tend to perform poorly in the presence of significant noise or complex degradation patterns. These limitations have motivated the adoption of learning-based approaches, where the model learns from data how to best distinguish between text and background under a wide range of conditions.

From a perceptual point of view, binarization is a visual enhancement process. It aims to improve the contrast between meaningful content and distracting background noise, rendering the document more legible to human readers. This aspect is particularly important in the context of historical manuscripts, where readability must be restored without compromising the integrity of the original text structure. In this regard, the quality of binarization affects not only machine understanding but also the user experience of historians, archivists, and scholars who depend on the clarity of the digital reproductions.

Finally, from an application-oriented perspective, binarization serves as a gateway to intelligent document processing. A high-quality binarization output enhances the accuracy of subsequent tasks such as text line segmentation, character recognition, structural analysis, and semantic indexing. Therefore, robust binarization contributes significantly to the reliability and scalability of document digitization pipelines and archival systems.

In this dissertation, we propose a learning-based binarization system that leverages Convolutional Neural Networks (CNNs) to classify pixels in small image patches as foreground (text) or background (non-text). The approach is based on constructing a carefully curated dataset composed of representative samples extracted from a wide variety of degraded document images. Using dimensionality reduction (PCA), unsupervised clustering (K-Means), and manual refinement, we prepare training data that captures the diversity of degradation

patterns. We explore both pretrained CNN architectures—adapted through transfer learning—and a custom-designed CNN tailored to our specific task and data constraints.

The proposed model is trained in a supervised manner and evaluated through rigorous experimentation. Our goal is to achieve a balance between classification accuracy, model complexity, and generalization ability. Special attention is given to preventing overfitting, addressing class imbalance, and ensuring robustness across different types of documents and degradation scenarios.

This work aims to contribute a reliable and efficient solution to the binarization problem, offering improvements in both perceptual quality and computational performance. The system can be integrated into larger pipelines for historical document analysis, thus supporting research, preservation, and accessibility initiatives in digital humanities and archival science.

The structure of this dissertation is as follows:

- **Chapter 1** introduces the background of document image analysis and reviews existing binarization techniques.
- **Chapter 2** presents the theoretical foundations of machine learning and CNN-based classification.
- **Chapter 3** describes the proposed methodology, including data preparation, model design, and training strategies.
- **Chapter 4** details the implementation and experimental results, along with comparisons and discussions.

Chapter 1. Binarization of Degraded Documents

1. Introduction

Binarization plays a significant role in image analysis as it facilitates the removal of distortions and unwanted artifacts from images while preserving essential information. This technique is widely used across various image types, including medical images, to enhance clarity and accuracy for subsequent processing.

Binarization is particularly beneficial for document images, supporting tasks such as document analysis and character recognition. By removing noise and addressing issues related to clarity, binarization significantly enhances the quality and readability of the text.

Over the years, binarization has continued to face challenges in achieving optimal performance, especially when dealing with significant noise, fading, blurring, and variations in lighting and contrast. However, advancements in research and the emergence of deep learning, often considered a black box of technologies, have introduced powerful solutions. Techniques such as Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs) have expanded the range of binarization methods, leading to impressive accuracy and robustness.

This chapter presents an overview of binarization, covering its fundamental techniques, benchmark datasets, and evaluation metrics. It examines the evolution of binarization methods, from traditional thresholding approaches to modern deep learning-based techniques. Additionally, the chapter discusses widely used datasets that serve as benchmarks for assessing binarization methods and explores key evaluation metrics that measure their effectiveness.

The rest of the chapter is organized as follows: Section 2 defines key concepts, while Section 3 examines document image degradations. Section 4 explores binarization approaches, including classical, machine learning-based, and deep learning-based techniques. Section 5 presents existing datasets, followed by Section 6, which discusses evaluation measures. Finally, Section 7 concludes the chapter.

2. Definitions

2.1. Image

The concept of an image has been defined in various ways across historical and modern literature.

Plato provided one of the earliest definitions, stating: “I call an image first the shadows, then the reflections seen in water or on the surface of opaque, polished, and shiny bodies, as well as all similar representations” [1].

In modern definitions, an image is often described in terms of its mathematical representation:

Benchamardimath and Hegadi [2] define an image as “an as an array, or a matrix, of square pixels (elements of picture) arranged in rows and columns”.

Johnson et al. [3] describe an image as “a spatially dependent intensity structure, the structure of which can be described in terms of an array of pixels”.

Obulesu and Kishore [4] define an image as “a two-dimensional light intensity function *f(x,y)*, where xx and *y* are spatial coordinates, and the function value at any pair of coordinates *(x,y)* is called intensity or gray level”.

Tran et al. [5] provide a similar definition: “An image may be defined as a two-dimensional function *f(x,y)* where *x* and *y* represent the spatial coordinates, and *f* represents the amplitude at any given pair of coordinates *(x, y)*. The amplitude is also often referred to as the gray level or the intensity of that point in the image”.

Dwivedi [6] states: “An image may be defined as a two-dimensional function, *f(x,y)*, where *x* and *y* are spatial (plane) coordinates, and the amplitude of any pair of coordinates *(x,y)* is called the intensity or gray level of the image at that point”.

Roy, Mitra, and Setua [7] define an image as “a finite set of spatial coordinates with attributes. It is described by geometric shapes and their relations using coordinates. Each coordinate is represented by a two-dimensional function *f(x,y)*, where the function returns a value known as intensity. Each coordinate holds data in various forms such as binary, grayscale, or color”.

2.2. Document Image

A document image is a specialized type of image that represents textual and graphical content in a digital format. The International Organization for Standardization (ISO) defines a document as:

“A combination of an information medium and the data recorded on it in a generally permanent and human- or machine-readable form”.

O’Gorman and Kasturi [8] further define a document image as:

"A digital representation of a document obtained through scanning or photography. It is analyzed to extract texts and graphics using techniques such as optical character recognition and page layout analysis."

Other definitions emphasize different aspects of document images:

- Digital Representation: Sun et al. [9] describe a digital document image as “the digital imaging result of paper documents, typically containing text, figures, and tables. It can be stored electronically on various media and networks, allowing users to access and carry it easily”.
- Static Record of Transactions: Loo [10] defines a document image as “a static representation of a specific recorded instance of a transaction, which can exist in either hardcopy or softcopy format. The former requires a scanning process to convert it into an electronic format. Unlike most ASCII documents, its content is represented by a collection of pixels”.

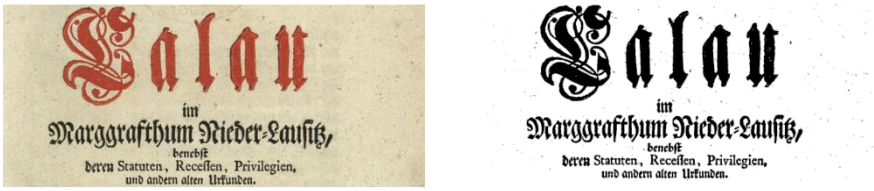
2.3. Binarization

Image binarization is the process of converting a digital image into a binary one, where each pixel has only two possible values: black or white.

As a preprocessing step in document analysis and recognition, binarization is one of the most fundamental and widely used segmentation methods. It separates the foreground text from the background in a document image, facilitating subsequent image processing tasks.

Binarization is also considered one of the simplest techniques for pixel classification, as indicated in [11]. In this approach, each pixel is classified as either foreground or background based on a comparison of its grayscale value with a predefined threshold.

Figure 1.1 illustrates an example of a binarization result, showing how the original image is transformed into a binary representation.



(a) **(b)**

Figure 1.1. Example of binarization, (a) original document, (b) its binary representation.

3. Types of document image degradations

Due to factors such as poor scanning, document aging, environmental conditions, and others, document images suffer from various distortions that hinder their processing. Among these distortions, we find:

a) **Uneven illumination:** Uneven illumination in document image analysis affects quality due to light absorption, particle diffusion, and background factors. This leads to artifacts during text extraction, especially in binarization, causing character recognition errors, notably in historical documents due to aging and poor storage (Figure 2.a) [12].

b) **Contrast variation: :** Contrast refers to the difference between high and low-intensity pixels within an object or between it and the background. In historical documents, noise, sunlight, and lighting affect image analysis and text extraction, especially when using thresholding to separate text from the background (Figure 2.b) [12].

c) Blur: Blur: blur is a challenge for algorithms attempting to restore sharp images. It softens images and blurs high-contrast details due to factors like camera movement (motion blur) or improper light convergence on the sensor (out-of-focus blur) as illustrated in Figure 2.c [13]

d) **Thin or weak text:** Poor writing quality in historical documents, due to faint ink or painted script, makes text difficult to read. Additionally, document characteristics add to the challenge. Researchers continuously develop strategies like tilt detection to enhance text extraction (figure 2.d) [14].

e) **Bleed-through degradation:** Ink bleeding, common in vintage fountain pen writings, causes smudging that blurs letters. This affects text extraction, particularly restoration and high-resolution image processing (Figure 2.e) [12].

f) **Faded ink or faint characters:** Document images degrade over time due to ink fading, tears, and holes, making text extraction difficult. Reduced contrast renders words illegible unless printed with high pressure (Figure 2.f) [15].

g) **Smears and spots**: Ink bleed-through in antique manuscripts causes ink to seep between pages, making text difficult to read and affecting text extraction. This degradation impacts both document appearance and readability (Figure 2.g) [12].

Figure 1.2. Examples of degradations in document images.

4. Approaches to document image binarization

Numerous techniques for image binarization have been proposed in the literature, with most focusing on grayscale images and fewer addressing color document foreground extraction. There is no universally accepted classification for document image binarization approaches, as categorization varies depending on the criteria used in different studies and the types of degradations present in document images.

Binarization methods are often categorized based on their approach, with one of the most common classifications distinguishing between global and local thresholding methods [16,17]. However, beyond thresholding, other classification schemes have been proposed. For instance, Kumar et al. [18] categorized binarization techniques into supervised methods (which use machine learning) and unsupervised methods (which rely on statistical information). Fernando

et al. [19] classified methods into thresholding-based and clustering-based approaches, with the latter further divided into region-based and classification-based techniques.

Sezgin and Sankur [20] introduced a classification system based on the type of information exploited by the methods, dividing them into six categories: histogram shape-based methods, clustering-based methods, entropy-based methods, object attribute-based methods, spatial binarization methods, and local methods.

More recent classifications, such as those proposed by [12,21], reflect the growing influence of machine learning and deep learning. These classifications typically distinguish between traditional methods (global, local, and hybrid thresholding) and deep learning-based methods, which leverage convolutional neural networks, generative adversarial networks, or attention mechanisms. Additionally, others organizes binarization methods into themes such as classical thresholding algorithms, image processing techniques, statistical models, and pixel classification, providing a structured overview of the field.

In this chapter, we adopt a three-category classification: Classical Methods, Machine-Learning based Methods, and Deep Learning-Based Methods, each of which can be further subdivided based on their specific characteristics and methodologies.

4.1. Classical binarization techniques

Traditional binarization methods for processing historical document images rely on image processing techniques to enhance text clarity and separate it from the background. These methods include global, local, and hybrid approaches, each utilizing thresholding to detect and extract text. In this section, we explore some of the most effective techniques.

4.1.1. Global thresholding

All methods in this category use a single global threshold to separate the foreground from the background in an image. The binarization process in global thresholding is performed by comparing each pixel's grayscale intensity x_i to a predetermined threshold T'. If the pixel intensity is greater than T', it is classified as foreground (white); otherwise, it is classified as background (black). Mathematically, this is expressed as:

$$bi = \begin{cases} 255, & if\ xi \leq T' \\ 0, & if\ xi > T' \end{cases}$$

Where bi is the binarized pixel value.

a) Fixed Global Threshold

It is the simplest global thresholding technique[22]. It involves comparing the grayscale value of each pixel to a predefined threshold T, such as 127.

b) Ostu's Method

Otsu's method is a global thresholding technique that aims to separate an image into two regions: the background and the foreground. The goal is to determine an optimal threshold T' that minimizes the overlap between these 2 classes by analyzing the grayscale histogram and maximizing the variance between them [23]. Mathematically, the optimal threshold is determined as:

$$T' = arg \max_{0 \le T \le L} [w_0(T) \cdot w_1(T) \cdot (\mu_0(T) - \mu_1(T))^2]$$

Where :

- T' : The optimal threshold.
- L : The total number of gray levels in the image.
- $w_0(T)\ and\ w_1(T)$: the probability of pixels belonging to the background and foreground, respectively.
- $\mu_0(T)\ and\ \mu_1(T)$: the mean grayscale intensities of the background and foreground, respectively.

c) Kapur's Method

Kapur's method, proposed by Kapur, Sahoo, and Wong, is a thresholding technique used for grayscale images based on the entropy of the histogram [24]. The method aims to determine an optimal threshold that maximizes the information between the object and background distributions in an image. The optimal threshold is calculated using the following steps:

Fist calculate Probability Distribution: $p_i = \frac{f_i}{N}$

Then entropy calculation : $H_s = -\sum_{i=1}^{s} p_i \ln p_i$ and $H_s = -\sum_{i=1}^{n} p_i \ln p_i$

After : $P_s = \sum_{i=1}^{s} p_i$

Then we calculate the of background : $H(A) = \ln P_s + \frac{H_s}{P_s}$

And the entropy for the foreground: $H(B) = \ln(1 - P_s) + \frac{H_n - H_s}{1 - P_s}$

After that we had threshold selection formula: $T = \arg\max_{s} H(A) + H(B)$

d) ISODATA Method

The ISODATA (Iterative Self-Organizing Data Analysis Technique) method follows the same general principle as the previous thresholding techniques but refines the threshold iteratively.

It begins by initializing the threshold, typically as the mean gray level of the image. The image histogram is then divided into two distinct classes: the background, which includes pixels with intensity values below or equal to the threshold, and the foreground, which consists of pixels with intensity values above the threshold. At each iteration, the method computes the mean intensity of both classes and updates the threshold as the average of these two means. This process is repeated iteratively until the difference between successive threshold values becomes negligible, ensuring convergence [25].

4.1.2. Local thresholding

Local thresholding techniques determine a threshold for each pixel based on its surrounding neighborhood. Instead of applying a single global threshold, these methods determine a local threshold *T(i,j)* dynamically using statistical properties of neighboring pixels. The binarization process is then performed as follows:

$$b(i,j) = \begin{cases} 255, & if\ x(i,j) \leq T(i,j) \\ 0, & if\ x(i,j) > T(i,j) \end{cases}$$

Where *b(i,j)* represents the binarized pixel value, *x(i,j)* is the original grayscale intensity of pixel (*i*,*j*), and *T(i,j)* is the local threshold.

a) Niblack's method

Niblack's method is a local thresholding technique that determines the binarization threshold for each pixel based on the statistical properties of its local neighborhood. It operates by sliding a window across the image and computing the local mean and standard deviation within each window [26]. The local threshold is calculated using the following formula:

$$T(i,j) = \mu(i,j) + k * \sigma(i,j)$$

Where:

- $\mu(i,j)$ is the mean of pixel intensities within the window centred over the pixel *x(i,j)*,
- $\sigma(i,j)$ is the standard deviation within the window centred over the pixel *x(i,j)*.
- *k* is a constant manually adjusted to determine threshold sensitivity

b) Sauvola and Pietikäinen's method

Sauvola and Pietikäinen's method is an enhancement of Niblack's algorithm, aiming to improve binarization in challenging conditions such as varying lighting and contrast. This technique calculates a local threshold for each pixel by considering the mean $\mu(i,j)$ and standard deviation $\sigma(i,j)$ of gray-level intensities within a window centered on the pixel [27]. The local threshold is determined using the following expression:

$$T(i,j) = \mu(i,j)\left(1 + k\left(\frac{\sigma(i,j)}{R} - 1\right)\right)$$

Here k is the user-defined parameter and *R* represents the maximum possible standard deviation of the pixel intensities, typically set to 128 for 8-bit images.

Sauvola's method improves resilience in challenging lighting and contrast conditions by using a lower threshold for windows with only background pixels, reducing the misclassification of background as foreground compared to Niblack's method.

c) Wolf's method

Wolf's method is an enhancement of Sauvola's binarization technique. This technique is mostly used on images that suffer from low contrast or a limited range of gray color intensities, as it adjusts the local statistics of the image based on the global statistics, thereby suppressing the deterioration of lighting and contrast levels [28].

The local threshold *T(i,j)* for a pixel at position *(i,j)* is calculated as follows:

$$T(i,j) = \mu(i,j) - k\left(1 - \frac{\sigma(i,j)}{S}\right)(\mu(i,j) - M)$$

Where:

- $S = \max_{ij} \sigma(i,j)$ is the maximum standard deviation across all local windows in the image,
- $M = \min_{ij} \mu(i,j)$ is the minimum mean intensity of all local windows,
- k is a user-defined parameter that controls the sensitivity of the thresholding.

d) Bataineh's method

Bataineh's method introduces an adaptive local binarization technique designed to address limitations in earlier methods like those proposed by Niblack and Sauvola. This approach dynamically calculates thresholds by considering both global and local image statistics, enhancing its ability to handle various degradations in document images.

The method begins by dividing the image into sub-regions or windows. For each window, it computes the local mean and standard deviation of pixel intensities. These local statistics are then combined with global image statistics to determine an adaptive threshold for each window. This integration allows the method to adjust to varying image conditions, such as low contrast or uneven illumination.

A key feature of Bataineh's method is its use of dynamic window sizing. The size of the window is adjusted based on the content within each region, enabling more precise thresholding. This adaptability helps in effectively separating foreground text from the background, even in degraded or complex document images [29].

e) Bernsen's method

Although it is a local thresholding approach, Bernsen's technique differs from the others by adjusting to variations in contrast or brightness across regions of varying sizes. It determines

the local threshold for each pixel by evaluating the local contrast within a defined window. Specifically, the local threshold *T(i,j)* is computed as the average of the maximum (I_{max}) and minimum (I_{min}) gray-level values within an $n \times n$ window:

$$T(i,j) = \frac{I_{\max} + I_{min}}{2}$$

If the local contrast, defined as I_{max} - I_{min}, is below a predefined threshold L, the area is considered to have insufficient contrast, and a default threshold is applied. Otherwise, the calculated *T(i,j)* is used to binarize the pixel [30].

f) Nick method

Nick method is a local thresholding technique derived from Niblack's algorithm, designed to improve binarization performance, particularly for degraded or low-contrast document images. In this method the local threshold is computed as:

$$T(i,j) = \mu(i,j) + k.\frac{\sqrt{\sum_{i=1}^{NP} p_i^2 - \mu(i,j)^2}}{NP}$$

Where p_i represents the intensity of the i^{th} pixel within the window, and *NP* denotes the total number of pixels in the window [16].

4.1.3. Hybrid thresholding

Hybrid thresholding combines the strengths of both classical and modern techniques to enhance text extraction in degraded document images. These methods integrate global thresholding techniques, such as Otsu's method, with local approaches, like Niblack's method, to achieve improved binarization results.

a) Saddami's Method

Saddami's method is an adaptive thresholding technique designed to enhance the binarization of ancient and degraded documents. This method combines three methods, two from local thresholding (Niblack and Wolf), and one global thresholding (Otsu's method) [31]. The threshold *T* is given by the expression :

When combine Niblack and Wolf

$$T_b = \frac{T_{Niblack} + T_{wolf}}{2}$$

When combine local and global threshold :

$$T_{clg} = \frac{T_b + T_{Ostu} + SD}{2}$$

Where $SD = \frac{\sigma}{2}$ is an adjustment based on the standard deviation of the pixel intensities.

b) Zemouri's Method

Zemouri's method is a hybrid binarization technique that combines global and local thresholding approaches to enhance the quality of historical document images. Initially, the method applies a global thresholding technique, such as Otsu's method, to the entire document to achieve a broad segmentation. Subsequently, it employs a local thresholding method, like Sauvola's technique, to refine the binarization at the pixel level [32].

c) Ntirogiannis and Gatos's Method

This technique combines global and local thresholding to enhance the binarization of handwritten document images. The method begins by applying Niblack's local thresholding to estimate the background and normalize the image, effectively handling local variations in illumination and contrast. Following this, Otsu's method is employed to remove small connected components and minimize ink blots, thereby refining the binarization result [33].

4.1.4. Advantages and Limitations of Traditional Binarization Methods

Traditional binarization methods are fast and simple to implement, providing a foundation for text extraction. However, each method has its strengths and weaknesses, as they often struggle with various types of document degradation, requiring more advanced techniques for improved accuracy. Table 1.1 provides a comparison of these methods.

Table 1.1. Comparison of Thresholding Methods for Document Image Binarization

Method	Advantages	Limitations
Fixed Global Thresholding	- Simple and fast - Effective for images with uniform lighting	- Ineffective for images with varying background intensities - Not suitable for degraded documents
Otsu's Method	- Automatically determines optimal threshold - Suitable for bimodal histograms	- Assumes bimodal distribution; may fail with overlapping intensities - Sensitive to noise and uneven lighting
Kapur's Method	- Utilizes entropy for threshold selection - More robust in certain scenarios	- Computationally intensive - Requires fine-tuning for optimal performance
ISODATA Method	- Iteratively adapts threshold - Can handle some intensity variations	- Slower due to multiple iterations - May not perform well on highly degraded images
Niblack's Method	- Effective under uneven illumination - Suitable for handwritten text	- Highly sensitive to noise - May introduce artifacts in background regions; requires careful parameter tuning
Sauvola's Method	- Reduces noise compared to Niblack - Performs well on degraded text	- Computationally demanding - Needs parameter tuning; may misclassify text in complex backgrounds
Wolf's Method	- Better for low-contrast images - Dynamically adjusts local statistics	- More computationally intensive - Sensitive to parameter selection; may introduce artifacts in bright backgrounds
Bernsen's Method	- Adapts to regions with varying contrast - Effective for documents with contrast variations	- Fails in low-contrast areas - May misclassify weak-edged text; not effective for noisy or historical images
Nick's Method	- Improved noise handling over Niblack - More stable performance	- Computationally intensive - Requires parameter tuning; limited effectiveness on highly degraded images
Bataineh's Method	- More adaptive than previous local methods - Incorporates global image statistics for accuracy	- Computationally expensive - Involves multiple processing steps; not widely adopted or standardized
Hybrid Methods (e.g., Saddami, Zemouri, Ntirogiannis and Gatos)	- Combine strengths of multiple techniques - More adaptable to varying conditions	- Computationally intensive - Increased complexity; requires balancing between local and global methods

4.2. Machine Learning-Based Binarization techniques

Traditional methods have suffered from several problems, especially documents with blurry backgrounds that are annoying and disorganized. To overcome these limitations, machine learning-based approaches have been developed, offering more robust solutions.

Many techniques based on machine learning have been developed to surpass the limits set by traditional methods. We will present in this section some of them.

4.2.1. Binarization methods based on SVM

Support Vector Machine (SVM) is a supervised learning algorithm used for image classification. In binarization, SVM utilizes key features to classify each pixel as either a background pixel or a text pixel. this means that SVM determines whether a given pixel belongs to the background or text class.

One of the proposed methods is that of Kita and Wakahara [34]. This method integrates k-means clustering with SVM and it consists of three stages: Firstly, K-means clustering groups pixels in the color space. Secondly, SVM verifies whether the classified regions contain actual text, and finally, SVM selects the optimal binary image. While this approach improves accuracy, it requires careful parameter tuning, and the chosen features significantly impact the results.

Another method, proposed by Chou, Lin, and Chang [35], divides the image into several regions, where SVM determines the best binarization strategy for each region. The decision is based on three key characteristics: Otsu's threshold, mean, and standard deviation. This approach improves binarization accuracy by adapting to variations within different image regions.

4.2.2. Binarization methods based on ANN

Artificial Neural Networks (ANNs) have gained significant attention in document image binarization due to their ability to learn complex patterns and adapt to various types of document degradation. Several ANN-based binarization approaches have been proposed, each utilizing different architectures and learning strategies.

One approach, proposed by Chen and Takagi [36] determines an optimal global threshold using an MLP trained with a modified backpropagation algorithm. The technique begins by computing the histogram of the contrast-enhanced image. The histogram is then scaled to the [0,1] range and divided into 128 parts. The MLP used consists of an input layer with 128 neurons receiving the 128 normalized histogram segments, two hidden layers with 256 and 512 neurons, respectively, and an output layer with 128 neurons.

In [37], the authors combined a global thresholding method, specifically Mass-Difference (MD) thresholding, with a supervised neural network (NN) to select the most optimal global threshold value. Each training image is first subdivided into multiple 32×32 pixel sub-images. Then, the MD algorithm is applied to each sub-image, producing several local threshold values, and to the entire image, resulting in a single global threshold. The supervised NN is then trained using the local threshold values as inputs and the global threshold value as the output. The trained NN is subsequently used to determine the most optimal global threshold for other images.

Lázaro et al. [38] proposed obtaining an optimal threshold for each image using a general regression neural network (GRNN) and a semantic description of its histogram. First, the input image's histogram is smoothed to eliminate false minima and maxima, and its discrete derivative is computed. Using a polygonal version of the derivative and the smoothed histogram, a new histogram description is calculated, from which a semantic description is inferred. The semantic descriptions of the training images serve as inputs to the neural network, while the corresponding optimal threshold values represent the desired outputs.

Kefali, Sari, and Bahi [39], proposed classifying pixels in historical documents as either text or background using a multilayer perceptron (MLP) network. The model is trained on global and local features extracted from deteriorated document images alongside their corresponding reference images to learn text patterns. During testing, each pixel is classified by the network, producing a clear binary image. This method is notable for its high accuracy and strong generalization capabilities, making it well-suited for handling various types of document degradation.

4.3. Deep Learning-Based Approaches

Deep learning has revolutionized document image binarization by offering advanced methods to overcome the limitations of traditional techniques. Among these, Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs) have been prominently utilized.

4.3.1. CNN-Based binarization

CNNs use convolutional and max-pooling layers to extract features from grayscale images, making processing more efficient than RGB images. Filters detect patterns like edges and textures, classifying each pixel as text or background based on local window features [40].

The first approach discussed in this section is the method proposed by Vo et al. [41], which employs a supervised deep learning network designed to predict text pixels at multiple feature levels. This multi-level feature extraction framework enhances text-background separation and improves the accuracy of text extraction compared to traditional methods.

Another method, introduced by He and Schomaker [42], integrates CNNs with Otsu's algorithm to refine threshold selection using deep learning and feature extraction techniques. This combination enhances performance, flexibility, and accuracy in separating text from the background, particularly in challenging conditions such as uneven lighting or image noise.

Long et al. [43] proposed fully convolutional networks (FCNs), which classify images at the pixel level and efficiently segment objects regardless of size. These networks process images of any dimensions by merging input features and reshaping them through convolutional and deconvolutional layers. Their approach aims to produce clear binary images with high efficiency while minimizing computational redundancy.

For instance, Tensmeyer and Martinez [44] developed a fully convolutional network (FCN) to handle binarization as a pixel-level classification task, feeding grayscale images and relative darkness features into a network with multi-scale encoding and upsampling layers to produce binarization masks.

In [45], an autoencoder model using convolutional networks was proposed. The model relies on encoding the image through convolutional and downsampling layers to extract mid-level representations, then decoding it again through those layers and upsampling to create a binary mask. After training the model, a global threshold is applied to obtain a clear binary image.

4.3.2. GAN-Based binarization

Generative Adversarial Networks (GANs) have emerged as a powerful deep learning approach for document image binarization, particularly in handling complex degradations such as shadows, noise, and uneven illumination. Several GAN-based methods have been proposed to address these challenges.

For instance, Suh et al. [46]. introduced a two-stage method for color document image enhancement and binarization using independent GANs. In the first stage, the focus is on removing unwanted patterns and extracting foreground information to improve image quality. The second stage involves separating each pixel into either a background or text pixel. This approach effectively addresses issues related to distracting backgrounds and colored noise.

Similarly, Bhunia et al. [47]. employed a game-theoretic approach and unsupervised learning to process deteriorated documents. They utilized a k-means algorithm to classify pixels into text and important elements or unimportant areas. To improve the quality of the conversion, they applied preprocessing to enhance the image and post-processing to correct errors. This method enhances the robustness of document binarization against various types of degradations.

De et al. [48]. proposed a dual discriminator generative adversarial network model called DD-GAN, which primarily relies on focal loss—a loss function designed to improve the class balance problem between pixels. It features two discriminators: a global discriminator for high-level features like texture and background, and a local discriminator that focuses on low-level details. This design allows the model to effectively handle various types of document degradations by ensuring that both overall structure and fine details are accurately binarized.

4.3.3. RNN-Based binarization

Recurrent neural networks (RNN) are a type of neural network characterized by processing sequential data. A notable approach exploiting an advanced type of RNN, which is Long Short-Term Memory Networks (LSTM Grid) is described here.

In [49], the image is firstly divided into small blocks consisting of a group of pixels, for example, 64×64, which makes it easier to convert the image into binary form. Next, a Grid LSTM is used to process each block of the image. This means that it processes each pixel within the block. However, it doesn’t stop there; the Grid LSTM also processes the adjacent blocks above, below, to the right, or to the left. This approach helps achieve a better contextual interpretation of the data. Then, the RNN reads the blocks sequentially, allowing the network to remember previous inputs when processing the current block. This sequential nature helps in understanding the relationships between distant pixels. Once all the blocks have been processed, the final results are merged to obtain the final binary image.

4.3.4. Challenges and Data Requirements

Table 1.2 provides a comparative analysis between deep learning models used for document images binarization. It hightlights the key challenges and data requirements of each model.

Table 1.2. Challenges and Data Requirements for Deep Learning Models in Document Image Binarization

Model	Challenges	Data Requirements
CNN (Convolutional Neural Networks)	- Requires large, well-annotated datasets to prevent overfitting. - High computational cost (requires powerful GPUs). - Sensitive to noise and complex document degradations. - Critical choice of hyperparameters (kernel size, number of filters, pooling size, etc.).	- High-quality, well-labeled data. - Diverse datasets to ensure good generalization. - Large dataset volume for optimal performance.
GAN (Generative Adversarial Networks)	- Training instability (mode collapse, gradient vanishing). - Difficult to control the quality and diversity of generated outputs. - Risk of imbalance between the generator and discriminator. - Strong dependence on the quality of input data.	- Large and diverse datasets to capture all possible variations. - High-resolution, well-labeled images for realistic generation. - Can use either labeled or unlabeled data, depending on the approach.
RNN (Recurrent Neural Networks)	- Difficulty in capturing long-term dependencies (partially addressed with LSTM/GRU). - Long training time due to sequential data processing. - Prone to overfitting, especially with limited data. - Complexity in tuning hyperparameters (number of layers, memory size, etc.).	- Well-structured sequential data. - High-quality data with precise annotations. - Datasets covering various degradations to improve model robustness.

5. Datasets

Several datasets have been developed to facilitate the evaluation and training of binarization algorithms. These datasets vary in size, content type, and complexity, providing diverse challenges for binarization techniques. All these datasets contain images with their corresponding ground truth binary images, enabling objective assessment of algorithm performance.

Below is an overview of some notable datasets:

5.1. DIBCO and H-DIBCO datasets

These datasets are the are pivotal resources in the field of document image binarization. They are proposed in the context of DIBCO (Document Image Binarization Contest) and H-DIBCO (Handwritten Document Image Binarization) competition series, organized from 2009 to 2019, with the exception of 2015 [50–59].

Each dataset comprises images with representational distortions from various and diverse libraries and collections in different languages such as Latin, Greek, etc.

5.2. PHIBD Dataset

The Persian Heritage Image Binarization Dataset (PHIBD) is a specialized dataset introduced in 2012 to support the evaluation and development of binarization algorithms, particularly for historical Persian manuscripts [60]. It comprises 15 images of historical and old manuscripts collected from the Documents and Old Manuscripts Treasury of Mirza Mohammad Kazemaini's library in Yazd, Iran. These images exhibit various forms of degradation, such as ink bleed-through, stains, and faded text, presenting significant challenges for binarization techniques.

5.3. CMATERdb 6

A benchmark dataset used to evaluate algorithms for converting document images to binary images. This dataset includes five color images representing a variety of documents, including documents captured by camera, others scanned, as well as modern and ancient manuscripts, some in poor condition and others in good condition. This dataset was presented in the research published by Ayatullah Faruk Mollah, Subhadip Basu, and Mita Nasipuri [61].

5.4. Bickley Diary Dataset

This collection consists of 7 images taken from personal diaries dating back to the 1920s. Its purpose is to support studies that preserve historical documents. These images exhibit the following deteriorations: paper damage, ink fading, and stains, all due to aging. They have been used in several research studies but are no longer available on the platforms [62].

5.5. ISOS (Irish Script on Screen) Dataset

A dataset dedicated to studying the problem of ink bleed-through in historical documents, consisting of 25 pairs of images recorded from the front and back of ancient Irish manuscripts, which are useful in evaluating the performance of document processing algorithms [63].

6. Evaluation measures

Evaluation metrics are used to determine the performance of techniques for converting documentary images to binary. This is done by comparing the outputs of any model with the reference image. These outputs measure accuracy and quality and provide a clear view of the efficiency of the algorithms.

6.1. F-Measure

The F-Measure, also known as the F1-Score, is a widely used evaluation metric in document image binarization [12]. It measures the balance between precision and recall as follows:

$$F - Measure = \frac{2 * recall * precision}{recall + precision}$$

Where,

$$recall = \frac{TP}{TP + FN}$$

$$precision = \frac{TP}{TP + FP}$$

- *TP* : The number of positive cases that were correctly classified
- *FN* : The number of positive cases that were incorrectly classified
- *FN* : The number of cases that were classified as positive but are actually negative.

In the context of document image binarization, a higher F1-Score indicates that the algorithm effectively distinguishes text from the background.

6.2. Peak Signal-to-Noise Ratio (PSNR)

It is a widely used metric for evaluating the quality of reconstructed or compressed images by comparing them to their reference counterparts.

A higher PSNR value indicates that the reconstructed image closely resembles the original, signifying better quality. PSNR is given by:

$$PSNR = 10 * log_{10}\left(\frac{{MAX_I}^2}{MSE}\right)$$

Where: MAX_I represents the maximum possible pixel value of the image, and *MSE* is the Mean Squared Error between the original and the reconstructed image, calculated as:

$$MSE = \frac{\sum_{x=1}^{M}\sum_{y=1}^{N}(I_{bin}(x,y) - I_{GT}(x,y))^2}{MN}$$

Here, *M* and *N* are the dimensions of the images, I_{bin} represente the pixel value after binarization, and I_{GT} is pixel value in the reference (ground truth) image [21].

6.3. Distance Reciprocal Distortion Metric (DRD)

The Distance Reciprocal Distortion (DRD) metric is designed to evaluate the visual quality of binarized document images by considering the impact of individual pixel errors on human readability [64]. DRD is calculated using the following formula :

$$DRD = \frac{\sum_{k=1}^{N} DRD_k}{NUBN}$$

Where DRD_k denotes the distortion of the k^{th} flipped pixel, and *NUBN* is the number of non-uniform color blocks in the reference (Ground Truth) image.

The lower its value, the higher the quality of the resulting image from the binarization.

6.4. Negative Rate Metric (NRM)

The Negative Rate Metric (NRM) quantifies the proportion of misclassified pixels in a binary image compared to a reference ground truth image [64]. NRM is calculate by :

$$NRM = \frac{NR_{FN} + NR_{FP}}{2}$$

$$NR_{FN} = \frac{N_{FN}}{N_{FN} + N_{TP}}$$

$$NF_{RP} = \frac{N_{FP}}{N_{FP} + N_{TN}}$$

Where N_{TP} is the number of true positives , N_{FP} is the number of false positives, N_{TN} is the number of true negatives, and N_{FN} is the number of false negatives.

The lower the NRM value, the higher the accuracy in classification.

7. Conclusion

In this chapter, we have explored the critical role of binarization in document analysis and image processing, emphasizing its importance in converting degraded or aged documents into binary form—transforming backgrounds to white and text to black—to enhance readability and facilitate further processing.

We have reviewed a variety of binarization techniques that have evolved over the years, each with its own advantages, disadvantages, and limitations. Traditional methods, such as global, local, and hybrid thresholding, have been widely used but often struggle with various distortions present in degraded documents. To address these challenges, newer approaches leveraging machine learning and deep learning have emerged, offering improved performance in handling complex degradations.

Our discussion encompassed both traditional and contemporary methods, highlighting their respective strengths and weaknesses. It is evident that the choice of binarization technique is crucial and should be guided by the specific requirements of the application and the characteristics of the document images in question.

In conclusion, while significant advancements have been made in binarization techniques, selecting the appropriate method remains a nuanced decision that must consider the unique challenges presented by each document image. Ongoing research and development in this field continue to enhance our ability to process and analyze a wide array of document types effectively.

Chapter 2. Foundations of Machine Learning and Associated Techniques

1. Introduction

In our current era, artificial intelligence and machine learning have shown great effectiveness in technological advancement, playing a significant role in facilitating and improving its quality and driving innovation through various real-life applications such as natural language processing, computer vision, medical diagnosis, and more. These technologies have gained widespread fame due to the vast availability of data and the abundance of details, enabling algorithms to learn and evolve in an unprecedented manner.

Among the most prominent applications of machine learning, we find image analysis and processing, where high-precision techniques are used for classification, segmentation, and restoration of distorted and degraded images. These technologies have opened new doors in many fields due to their ability to simulate or even surpass human perception.

One of the most important branches of machine learning is deep learning, which relies on multi-layer neural networks capable of efficiently processing complex data such as images, videos, and texts. Deep learning excels in many tasks involving fine patterns and numerous details.

This chapter provides a comprehensive foundation on the fundamental concepts and techniques of machine learning, particularly those relevant to image processing tasks. We begin by outlining the general principles of machine learning, including its main types. The discussion then progresses to deep learning, a subset of machine learning characterized by hierarchical neural network architectures capable of modeling complex patterns in data. In addition, unsupervised learning methods such as clustering, with a focus on the K-Means algorithm, are presented to highlight approaches for grouping unlabeled data. Finally, the chapter covers essential data preprocessing techniques, including dimensionality reduction methods like Principal Component Analysis (PCA).

2. Machine Learning : Concepts

2.1. General Overview

Machine learning is a branch of artificial intelligence that refers to learning from data, or more accurately, it was defined by Minsky as a set of computational procedures that operate automatically to learn from examples without human intervention during operation. The goal of machine learning is to produce classification models that are understandable to humans and to mimic or surpass human thinking in some way [65].

2.2. Application in Image Processing

There are many applications in image processing, and we mention some of them [66]:

- ***Image classification:*** This involves assigning a label to an image, such as identifying whether it contains a cat or a dog. It is widely used in fields like healthcare and security.
- ***Object Detection:*** Beyond classification, object detection identifies specific objects in an image and locates them with bounding boxes. It is essential in autonomous driving, surveillance, and robotics.
- ***Image Segmentation:*** This task labels each pixel of an image, enabling fine-grained analysis. Semantic segmentation assigns categories to pixels, while instance segmentation

also distinguishes between objects of the same type. It's used in medicine and autonomous systems.

- ***Image Enhancement:*** Aimed at improving image quality, especially in degraded or low-light conditions. It is useful in restoring historical documents or preparing images for further analysis.

2.3. Types Of Machine Learning

Machine learning techniques are generally categorized into three major types, based on the nature of the learning process and the data involved:

2.3.1. Supervised Learning

Supervised learning is considered one of the branches of machine learning, with its main foundation being the presence of labeled training data, meaning that the inputs and outputs contained in the system are known. The goal of this type of learning is to build a model capable of predicting correct outputs when new data is input [67].

Supervised learning consists of two main stages :

- **Training phase** : where the model is given a set of data containing labels and features for the model to learn the relationship between them.
- **Testing phase** : We provide the model with new data it has not seen before to evaluate it.

This type of learning is also divided into two main categories :

- **Classification** : It is used when the outputs are specific categories or labels, such as classifying emails as important or not important.
- **Regression** : The outputs here are continuous numerical values, such as predicting the price of a piece of land based on a set of characteristics.

2.3.2. Unsupervised Learning

Unsupervised learning is one of the main branches of machine learning. Unlike supervised learning, which relies on labeled data with known outputs, unsupervised learning works with unlabeled data, meaning the model does not have predefined target variables. The primary goal is to discover hidden patterns**,** structures**,** or relationships within the input data.

Several key techniques are commonly used in unsupervised learning [68]:

- **Clustering:** This technique groups data into clusters based on similarity, where items within the same cluster are more alike than those in different clusters. Algorithms such as K-means and DBSCAN (density-based spatial clustering of applications with noise) are widely used in this context.
- **Dimensionality Reduction:** This involves reducing the number of input variables while retaining as much relevant information as possible. It is often used to simplify data visualization and speed up analysis. Common methods include Principal Component Analysis (PCA) and T-distributed Stochastic Neighbor Embedding (t-SNE).

- **Association Rule Learning:** This method identifies interesting relationships or correlations between variables in large datasets. It is frequently applied in market basket analysis to uncover purchasing patterns.

Despite its strengths in revealing unexpected insights and structures within data, unsupervised learning presents challenges. The lack of ground truth makes it difficult to evaluate model performance objectively. Additionally, methods like clustering may produce ambiguous or misleading results depending on how clusters are defined or the number selected [69].

2.3.3. Reinforcement Learning

Reinforcement learning is a type of machine learning that relies on an agent, which starts in a certain state and chooses a specific action in a certain environment. The environment responds with a new state and a reward, which could be points or an evaluation. The agent then tries to choose actions that help it earn the maximum possible reward in the long term.

There are techniques used in reinforcement learning, and we mention three of them:

- **The balance between exploration and exploitation** : Exploration means that the agent tries new actions, while exploitation is the use of actions. It rewards him with good rewards.
- **Q-learning :** It is a widely used method for learning the best action in every state.
- **Deep Q-Networks (DQN):** also known as deep Q-learning, function like traditional Q-learning, but with a key difference: they use neural networks to approximate the Q-values.

Reinforcement learning is characterized by its ability to learn from experience and adapt to changing environments without pre-existing data, but at the same time, it faces challenges such as slow learning and difficulty in achieving balance [70].

3. Deep Learning

3.1. Definition and positioning of deep learning within machine learning

Deep learning is a branch of machine learning [71]. It involves a set of algorithms that use multilayer neural networks to automatically and gradually learn and represent complex patterns within raw data. Traditional machine learning relies on manual feature extraction, but deep learning models automatically extract features through successive layers. Their inclusion of many hidden layers enables them to achieve tremendous effectiveness with large and complex datasets across various fields. With their diverse techniques, such as convolutional neural networks, recurrent neural networks, and autoencoders, they have opened new horizons for artificial intelligence [72].

3.2. Main characteristics

Deep learning has achieved remarkable milestones in artificial intelligence, such as recognizing human activities and its ability to handle data imbalance issues, among others. All of this is due to its distinctive features [73] which we will mention :

1. Its reliance on deep neural networks, which are structures composed of multiple layers of neural networks.

2. It is characterized by a strong learning ability from large datasets, even unclassified ones, and the discovery of complex patterns.
3. It is characterized by a strong ability to generalize even with little training data.
4. It surpasses human capabilities in complex computational tasks and feature extraction.
5. His performance accuracy is tied to the network architecture, the type of activation function, and the method of data representation.

3.3. Applications examples in Computer Vision

So, deep learning in computer vision is basically teaching computers to see and understand images, and it's used in a bunch of cool ways:

1. **Medical stuff:** Instead of doctors looking through tons of X-rays or MRIs, AI can help spot diseases like tuberculosis or find tumors. It's like having a really fast assistant pointing out the important bits.
2. **Robots and self-driving cars:** These use deep learning to "see" the road and obstacles. There's this system called YOLO that's really good at spotting cars and people quickly so the car knows where to drive safely.
3. **Describing images:** Some AI can actually look at a photo and write a caption about what's in it, or even answer questions about the picture — like a tiny smart storyteller.
4. **Other stuff:** Things like unlocking your phone with your face, recognizing traffic signs in autonomous cars, and turning handwritten notes into typed text — all powered by deep learning.

Basically, it's all about making computers smarter at understanding what they see to help us in real life [74].

4. Convolutional Neural Networks (CNN)

4.1. Introduction to CNNs

Convolutional neural networks first appeared in 1989 and were introduced by LeCun [75]. They are inspired by the way the visual cortex in the human brain works. CNNs are a special type of artificial neural networks inspired by the biological neural networks in the brains of living organisms. They aim to process data such as images and videos through automatic learning of their distinctive features. They are characterized by the use of interconnected and multiple neural layers that rely on the convolution process, which makes them different from traditional neural networks. They also consist of pooling layers and fully connected layers.

These layers work gradually to extract features incrementally from the data using the backpropagation property to improve the model, and each cell has a limited receptive field that focuses only on a specific part of the data, meaning that each cell, for example, focuses on a specific part of the image [76].

4.2. Basic concepts of Artificial Neural Networks

Artificial neural networks (ANNs) are computational models inspired by the human brain. Its primary goal is to simulate how the human brain processes information. The models of these

networks work on discovering patterns and relationships between data through learning. Its basic structure It consists of units known as artificial neurons that are interconnected with adjustable weights, and it has three organized layers (Figure 2.1):

- **Input layers:** receive raw data.
- **The hidden layers** process the data internally and extract features from it.
- **The output layer** gives the final result, whether it is classification or prediction.

Each neuron receives a set of weighted signals, then applies an activation function (such as ReLU or Sigmoid) to produce an output that is passed to the next neurons.

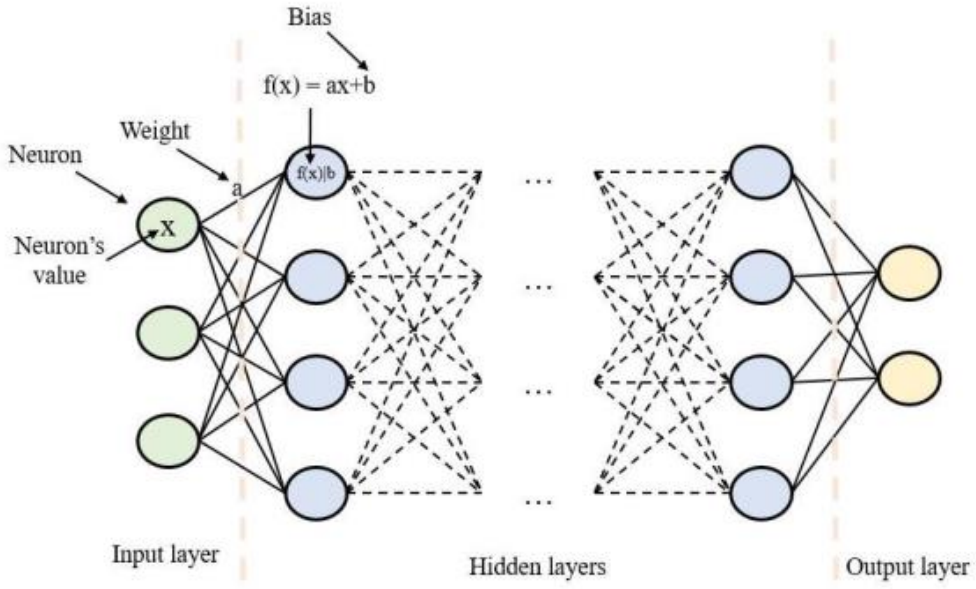


Figure 2.1. Basic architecture of an ANN.

4.3. Typical CNN Architecture

A typical CNN architecture is composed of several key layers that work together to extract features and perform classification or regression tasks. Figure 2.2 illustrates the basic structure of a CNN.

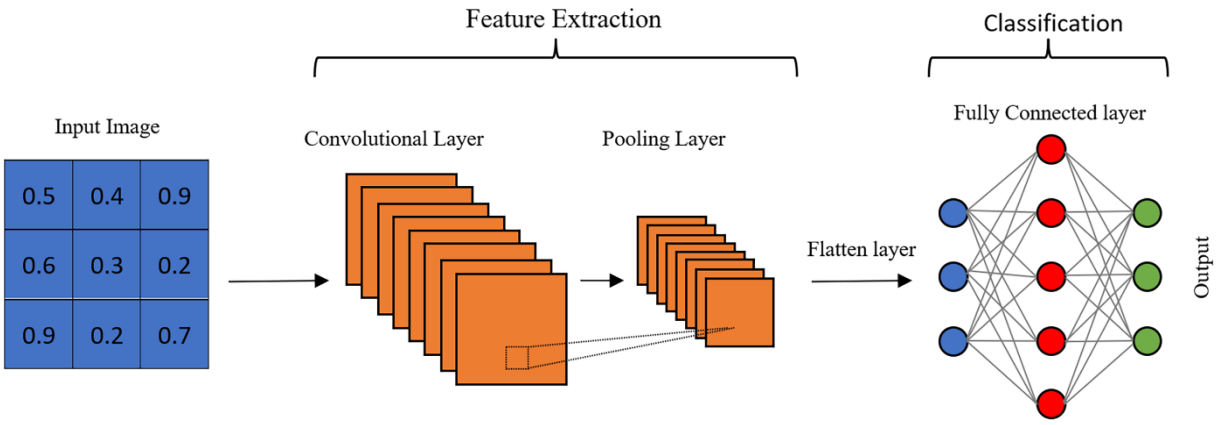


Figure 2.2. Architecture of a simple CNN [77]

4.3.1. Convolutional Layer

The convolutional layer is the distinctive element that sets convolutional neural networks apart, and it plays a fundamental role where most of the calculations are performed. This layer relies on a set of learnable filters or kernels, which are small in terms of spatial dimensions but extend through the full depth of the input data.

When the data enters and passes through this layer, each filter from the set of approved filters is passed through the dimensions of the image to extract a two-dimensional activation map to form a feature map. Then, these extracted maps are merged to form the final output size, where the cells within the same activation map share common parameters, meaning the same parameters, which helps reduce the number of parameters in the network and decrease complex calculations.

The output dimensions in the convolutional layer are determined by three main hyperparameters, which we summarize in the following table.

Table 2.1. The main hyperparameter of convolutional layer

Hyperparameter	Definition	Function and Effect	Example
Depth	Number of filters used in the convolutional layer	Determines the number of resulting feature maps. Each filter extracts a unique feature	Using 32 filters results in 32 feature maps
Stride	Number of pixels the filter moves across the image	The higher the value, the smaller the output. Used to progressively reduce dimensions	Stride = 1 → larger output, Stride = 2 → smaller output
Zero Padding	Adding zeros around the original image before convolution	Used to preserve the original image size or control the output size; prevents loss of edge information	Padding = 1 preserves dimensions with a 3×3 filter

During the training of the network, the backpropagation algorithm is used, which includes an error correction phase that is done by convolving the filters after flipping them to update the weights accurately.

Some modifications were made to the convolutional networks to address certain limitations and improve network performance in more complex tasks :

- Network within a network, its main idea is to use 1×1 filters, but inside them, there are multi-layered neural networks that help the model understand complex relationships within the data.
- The dilated convolution is a special type of convolution that relies on expanding the filter to cover a larger area of the image without losing precision. This modification was introduced to increase the amount of information the filter can see [78].

4.3.2. Pooling Layer

The previous convolutional layer produces feature maps, which are then sampled and their dimensions reduced through a process at the pooling layer. This process aims to reduce spatial dimensions while retaining essential information, or in other words, the dominant features. This process also requires determining the kernel size and its step before execution. There are many available methods for pooling, including tree pooling, directed pooling, minimum pooling, maximum pooling, and global average pooling. The last three methods are among the most commonly used in practical applications.

The pooling layer offers many specific advantages, such as reducing computational cost and achieving some stability against spatial transformations. However, its use can lead to the loss of some precise information, which may result in a decrease in model performance in certain cases. This is due to the layer's focus on detecting the presence of a feature without paying attention to its exact location, sometimes leading to the neglect of important spatial details [79].

4.3.3. Activation Functions

The main component in neural networks is the activation function. Its role is to convert the input into an output. This input is usually calculated by summing the weighted inputs of the neuron with a bias. Based on this value, the activation function decides whether the neuron will activate or not.

In convolutional neural networks, non-linear functions are applied after learnable layers such as convolutional or dense layers. Non-linearity adds the ability for the model to represent complex relationships, which must be differentiable in order to activate the backpropagation algorithm during training [79].

There are many activation functions used in CNNs, and among the most famous are:

- ***sigmoid:*** which accepts real numbers as input and returns an output restricted between 0 and 1, taking the shape of an S-curve. It can be expressed by the following function:

$$sigm(x) = \frac{1}{1 + e^{-x}}$$

- ***ReLU:*** This function is characterized by computational efficiency, which is why it is the most commonly used in the context of convolutional neural networks. It outputs zero for negative values and retains positive values. However, sometimes it can cause a problem where some neurons stop activating, known as dead ReLU. It can be expressed by the following function :

$$ReLu(x) = \max(0, x)$$

- ***Leaky ReLU:*** It is considered an alternative to ReLU, allowing a small portion of negative values to pass through, thus avoiding the "dying ReLU" problem.

$$LeakyReLU(x) = \begin{cases} x \; if \; x > 0 \\ mx, x \leq 0 \end{cases}$$

And *m* is a very small value such as 0.001.

- ***Softmax:*** This function is primarily used in the output layer for multi-class classification tasks. It transforms a vector of raw scores (logits) into a probability distribution over possible classes. Each output value is in the range [0,1]. It is defined as:

$$Softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

  Where z_i is the i^{th} *input value* (*logit*) to the softmax function, and *n* the number of classes.

The figure below shows the shapes of the most commonly used activation functions.

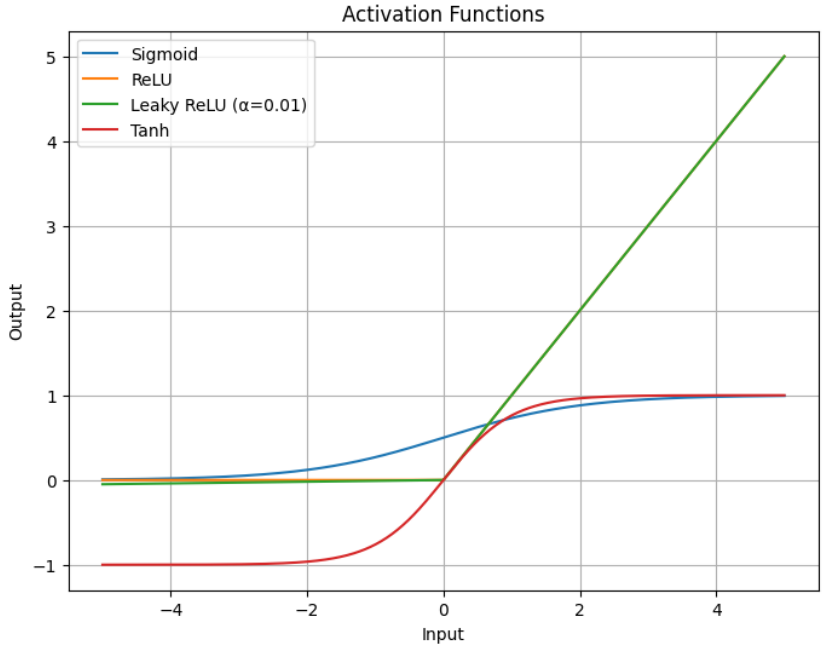


Figure 2.3. Graphical Comparison of Common Activation Functions

4.3.4. Fully Connected Layers

This layer is typically located at the end of a CNN and serves as the classifier within the network. It connects each neuron to all neurons in the previous layer, a structure known as a fully connected layer. It follows the same principle as traditional neural networks of the Multilayer Perceptron type and is considered a form of feedforward neural network. The input to this layer is a vector derived from the feature maps—flattened outputs from the final pooling or convolutional layer. The output of the fully connected layer represents the final prediction of the network [79].

4.4. Cost Function and Optimization

To understand how networks learn and improve their performance and capacity, it is better to study basic elements such as the loss function and optimization algorithms, which we will cover in the present section.

4.4.1. Loss Function

The loss function is a fundamental element in training CNN models as it measures the difference between predictions and actual results, providing a clear metric that we can use to evaluate the performance of any model. This function is used in the backpropagation algorithm

to adjust the model's weights in order to reduce error and improve the model's accuracy. It also balances bias and variance to avoid overfitting [80].

However, Different tasks and models require specific loss functions tailored to the nature of the problem. Here are some widely used loss functions:

a) Mean Squared Error (MSE) Loss Function

The mean squared error is also called the squared error loss or L2 loss. It is commonly used in regression problems, where it is calculated by averaging the squares of the differences between the actual and predicted values according to the following formula.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

So that *N* represents the number of samples and y_i is the true value , $\hat{y}_i$ is the predicted value. The advantage of this function is that it severely penalizes large errors due to squaring, which helps training algorithms reduce large errors and improve the model's accuracy. Additionally, the value of this function is always positive and approaches zero as the model's prediction accuracy increases. However, it is sensitive to outliers due to their clear impact on the overall error value [81].

b) Binary Cross Entropy (BCE)

The function is also known as log loss and is one of the most famous loss functions used in binary classification problems. It measures the difference between the true labels (0 or 1) and the probabilities produced by the model, providing an indication of how well the predictions match the reality. The lower the value of the function, the better the model performs. We can define it mathematically with the following expression [82] .

$$BCE = \frac{1}{N}\sum_{i=1}^{N} y_i \log(p_i) + (1 - y_i)\log\,(1 - p_i)]$$

In the field of pixel-level image classification, such as background and text segmentation in document images, the function is used to evaluate the model's prediction accuracy for each pixel individually. It measures the difference between the predicted probabilities and the actual binary labels for each pixel [83].

c) Categorical Cross-Entropy (CCE)

The function is one of the most commonly used functions in training deep neural networks, especially in multi-class classification tasks. This function relies on measuring the difference between two probability distributions: the first represents the true distribution of the class, and the second represents the probability distribution resulting from the model after applying the softmax function. It can be expressed by the following mathematical expression :

$$CCE(p, q) = p(x_i) \log q(x_i)$$

For $p(x_i)$ is The actual distribution and $q(x_i)$ is the distribution resulting from the model.

This function is very useful in measuring the alignment of the model's probabilistic expectations with the correct classes. However, its use in cases with noise in the data or labels can lead to overfitting, which weakens the model's ability to generalize later [84].

4.4.2. Optimization Algorithms

Optimization algorithms are fundamental components in training neural networks, responsible for updating the model parameters to minimize the loss function. These algorithms iteratively adjust weights and biases by computing gradients of the loss with respect to the parameters, guiding the network toward better performance. Here are the common optimizers:

a) Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent is one of the most widely used optimization algorithms in deep learning. It operates by updating the model parameters based on the gradient of the loss function, computed using small, randomly selected subsets (mini-batches) of the training data. In each iteration, an estimate of the gradient is calculated from the current mini-batch, and the parameters are updated accordingly using this estimate and a learning rate.

The learning rate is a critical hyperparameter that influences both the speed and stability of the learning process. If set too high, it may cause the model to overshoot optimal solutions, leading to oscillations or divergence. Conversely, if the learning rate is too low, the model may converge very slowly or get stuck in suboptimal minima.

The learning rate is typically chosen through experimentation or by employing techniques such as line search. Alternatively, adaptive strategies can be used to adjust the learning rate dynamically during training in order to improve convergence [85].

b) Adam

Adam (Adaptive Moment Estimation) is one of the most popular optimization algorithms used in deep learning. It computes adaptive learning rates for each parameter in the model by estimating two statistical moments of the gradients. The first moment corresponds to the exponentially weighted average of past gradients (similar to momentum), while the second moment corresponds to the exponentially weighted average of the squared gradients.

A key feature of Adam is that it integrates momentum by directly estimating the first moment and includes a mechanism to correct bias introduced by initializing the moment estimates at zero. Specifically, Adam first computes the moving averages of the gradients and their squares, and then applies bias-correction to these averages. These corrected estimates are subsequently used to update the model parameters.

Adam offers several advantages, including reduced sensitivity to the initial choice of learning rate, improved convergence speed, and ease of implementation. It is also considered to be a robust and stable optimizer, relatively insensitive to the scale of the gradients [85].

4.5. Pre-trained CNN models

Due to the difficulty of training CNN models from scratch and the high time and computational costs Pre-trained CNN models have emerged to facilitate training and processing, and they have

proven to be effective and considered a viable solution since they have been trained on large datasets. They can be reused or modified for various and diverse tasks, including:

4.5.1. Lenet

LeNet first appeared to perform the task of classifying handwritten digits, developed by Yann LeCun in 1998. It is considered one of the first and most influential models in the context of convolutional neural networks.

The model relies on a network of seven layers, including convolutional and pooling layers, and is primarily used for processing grayscale images of a specific size, which is 32×32 pixels.

The structure of LeNet relies on applying several convolution operations followed by several pooling operations before connecting the final convolution layer to a fully connected layer, ending at the output layer. Since the convolution and pooling layers do not use padding and operate with a stride of one, this can lead to a gradual reduction in the dimensions of the image after each layer it passes through, thereby shrinking the overall representation size. LeNet has been used in many applications as it is suitable for small-sized data. Among these applications is feature extraction in brain-computer interface systems, where the model was integrated with the common spatial pattern algorithm for signal analysis [86].

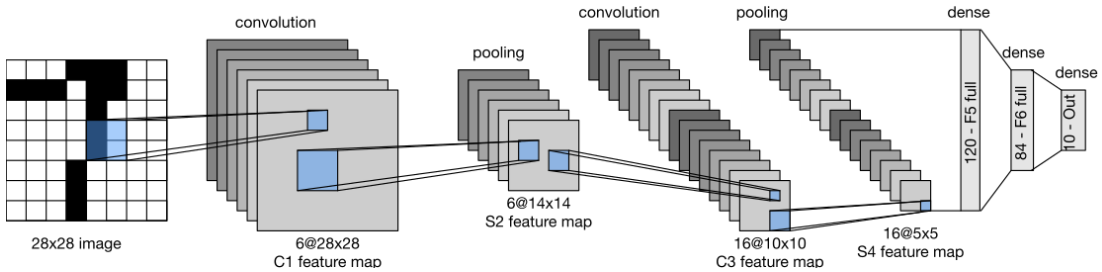


Figure 2.4. Illustration of lenet-5 architecture

4.5.2. VGG

Due to the effectiveness of convolutional neural networks in the field of image recognition and processing, researchers Simonyan and Zisserman presented a simple and efficient architectural model known as VGG, short for Visual Geometry Group.

This model is characterized by greater depth compared to other models because it contains a larger number of layers, reaching up to 19 layers in some configurations.

The VGG model relied on a series of small filters with dimensions 3×3 instead of larger filters due to some studies on other models that proved small filters improve network performance. Experimental results showed that stacking these small filters can achieve an effect similar to larger filters in terms of receptive field coverage, with the added benefit of reducing the number of parameters and thus lowering computational complexity.

The VGG model utilized 1×1 convolutional layers between traditional convolutional layers to reduce network complexity by learning linear combinations of feature maps. Additionally, it employed padding to preserve spatial dimensions and incorporated max pooling layers after each set of convolutional layers.

VGG achieved good results in image classification and localization thanks to its depth, but despite its effectiveness, it requires a large number of computations, making it computationally expensive [79].

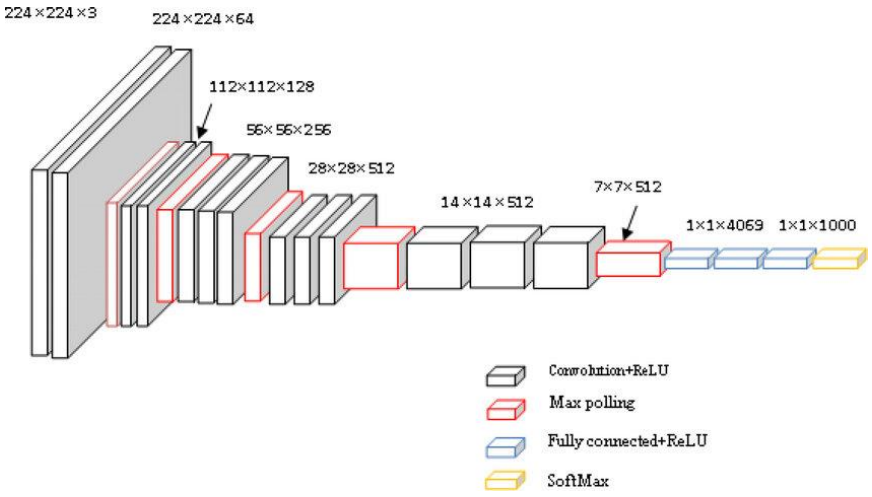


Figure 2.5. General Architecture of the VGG Network

4.5.3. MobileNet

After the lack of effective models for working on devices with limited resources, the MobileNet model was designed to be lightweight, accurate, and efficient by using depthwise separable convolutions, which reduces the number of parameters and computational complexity while maintaining good accuracy.

The model, like the VGG model, relies on small filters of size 3×3, takes input images of size 224×224, and uses the well-known ReLU activation function, along with the Adam optimization algorithm and a learning rate of 8×10^{-3}.

MobileNet is distinguished by its high training speed, achieving 90% accuracy in 180 seconds. It also excelled in simple tasks, such as classifying colored containers, achieving 100% accuracy after a limited number of epochs.

Despite its speed and accuracy, the model suffers from a decrease in performance when using small datasets. It is also less stable with more complex data and less accurate than some models in the case of complex tasks [87].

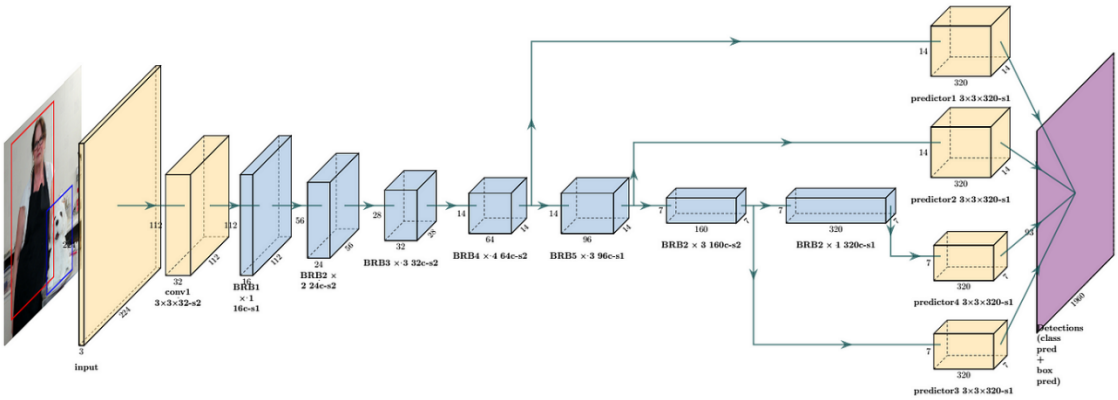


Figure 2.6. General Architecture of MobileNet

4.5.4. DenseNet

To address the problem of vanishing gradients, the DenseNet model was proposed as a solution to this issue by establishing improved connections between layers. It primarily relied on the principle of cross-connections between all layers using the forward propagation method, meaning that each layer's inputs come from all the previous layers.

DenseNet differs from traditional networks because the number of connections in it does not equal the number of layers; instead, it contains $\frac{l(l+1)}{2}$ where l is the number of layers in the model.

DenseNet is defined by its capacity to differentiate between newly added and retained information, as it amalgamates feature maps from preceding layers by concatenation rather than summation. This approach, nevertheless, results in an augmentation of feature maps, hence elevating the computational expense, notwithstanding the model's compact layer configuration. A primary feature of DenseNet is that all layers obtain gradients directly from the loss function, hence improving the information flow throughout the network. This direct connection also mitigates overfitting, particularly in tasks dependent on limited training datasets [79].

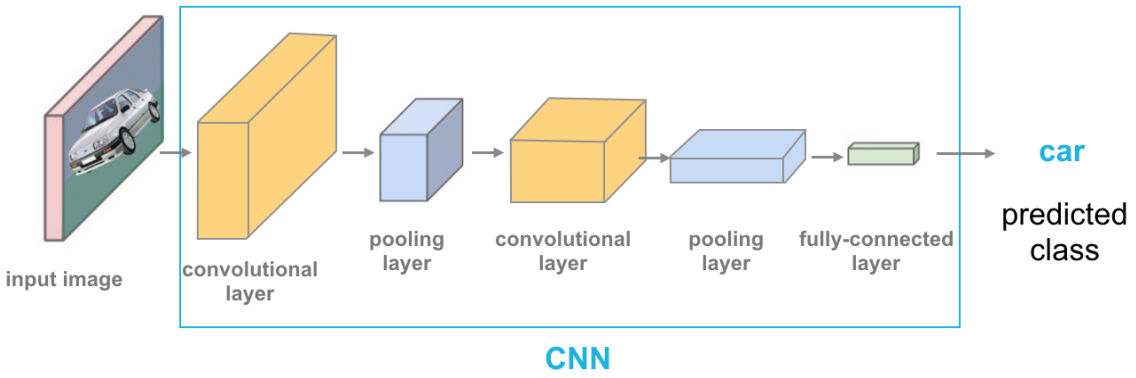


Figure 2.7. DenseNet architecture

4.6. Strategies to improve performance

To improve the performance of deep models, several strategies can be adopted. These techniques play a crucial role in improving training stability, accelerating convergence, and preventing overfitting.

Overfitting occurs in deep neural networks due to the large number of parameters and their high flexibility. When the model performs excellently on the training data but fails to generalize to the test data, it is because the model learns the patterns and noise specific to the training data, leading to a significant difference between training accuracy and test accuracy. And signs of overfitting include a decrease in training loss and an increase in validation loss [88]. Since every problem has a solution, techniques have emerged to prevent overfitting, including:

4.6.1. EarlyStopping

Early stopping is considered a simple technique despite its effectiveness in reducing the problem of overfitting in neural networks. This technique relies on splitting the data into a

training set and a validation set, then monitoring the error on the validation set during training. Once the error starts to increase, training is stopped, and the model weights are retained at the best previous point [89].

4.6.2. Dropout

It is a simple yet effective regularization technique used during the training of neural networks aimed at reducing overfitting. Its basic idea is to randomly drop some neurons during each training phase using a Bernoulli distribution, which forces the network not to overly rely on certain neurons, thereby learning more generalized representations [90].

4.6.3. Batch Normalization

Batch normalization is considered one of the prominent techniques for improving the performance of convolutional neural networks. It normalizes each channel over a small batch of data by subtracting the mean and dividing the values by the standard deviation. Where the resulting values are re-centered and scaled. Learnability, which leads to reducing the variation of internal distributions and contributes to making the training process faster and reducing the model's sensitivity to weight initialization.

Batch normalization enables the use of higher learning rates and helps regulate the input values to activation functions, improving the usability of functions like sigmoid or ReLU in deep networks. The batch normalization technique is also characterized by reducing reliance on random dropout because it adds a type of noise .

4.7. Relevance of CNNs for Image Binarization

CNNs have proven highly effective in binarizing degraded images, treating the task as semantic segmentation. Models like U-Net and FCNs excel in feature extraction and accuracy, especially when enhanced with multi-scale techniques. Overall, CNNs offer a powerful deep learning framework that outperforms traditional binarization methods.

4.7.1. Pixel-wise Learning

The task of segmentation is a type of semantic segmentation where each pixel in the images is classified into two categories: either text, which is black, or background, which is white. To achieve better results than traditional methods, convolutional neural networks have been adapted to models suitable for this type of task. These models have already proven their capability and effectiveness in accurately classifying pixels. Among the notable works are those by Tensmeyer, Martinez, Vo, and Peng, who developed FCN models capable of learning features from multiple levels [14].

5. Clustering

5.1. Introduction to Clustering

Clustering is a data analysis process that involves classifying data into groups based on the similarity of their characteristics. Many clustering techniques have emerged to solve various problems. Traditionally, clustering methods are classified into two main categories: partitioning clustering and hierarchical clustering as it is shown in Figure 2.8 [91]. Despite the efficiency achieved by these methods, they force researchers to input the number of clusters into the data,

which is not feasible in most practical cases. To overcome these limitations, the idea of automatic clustering techniques emerged, which aim to determine the number of clusters automatically without prior information about the nature of the data. Many studies in the literature have addressed these methods, applying them in various fields including statistics, computer science, and machine learning.

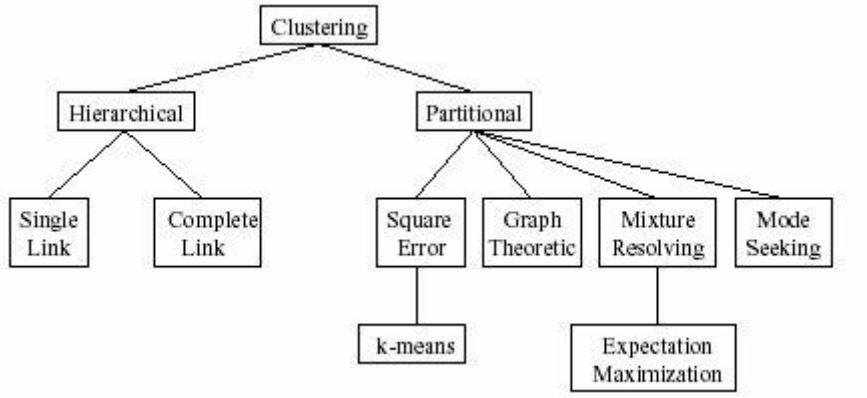


Figure 2.8. A taxonomy of clustering approaches [91]

Some studies have also indicated that no available clustering technique provides ideal performance for all types of large datasets, especially in genomic data [92].

5.2. K-means Algorithm

The K-means algorithm is the most commonly used and one of the most famous clustering algorithms. It is an iterative algorithm used to divide a set of *N* elements into a predetermined number of separate groups, *k*. The algorithm falls under centroid-based clustering methods, where each cluster is represented by the mean of its constituent points, also known as the centroid [93].

5.2.1. General Functioning

The K-means algorithm proceeds through the following key stages:

1. ***Determining the number of clusters k:*** In clustering techniques, the number of clusters must be known in advance before using the technique, and it usually requires several experiments to determine the best number.
2. ***Repetition:*** It goes through two main stages:
   a) *Assignment phase:* Each data point is assigned to the nearest cluster using Euclidean distance.
   b) *Centroid update phase:* where a new centroid is calculated for each cluster by computing the geometric mean of the points belonging to it.
3. ***Stopping criteria:*** The algorithm stops when the points no longer change clusters or when the maximum number of iterations is reached.
4. ***Measuring the quality criterion:*** or the quality of the partition by summing the squared errors within the clusters.

A visual representation of the K-means clustering process can be seen in the following figure.

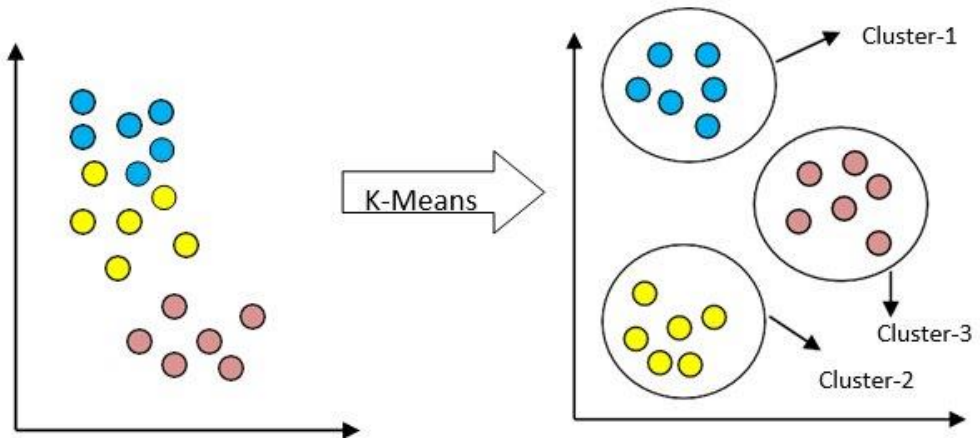


Figure 2.9. Clustering example using K-means

5.2.2. Initialization, Convergence, Choice of number of Clusters

a) Initialization

The initialization phase in the K-means algorithm is considered a critical stage as it significantly affects the quality of clustering and the speed of reaching the final solution [94] Poor initialization may lead the algorithm to suboptimal solutions or increase the number of iterations needed. Various methods have been developed to address this, differing in complexity and effectiveness. Table 2.2 presents a summary of some commonly used initialization techniques.

Table 2.2. Summary of Common K-means Initialization Methods

Type	Method	Summary
Random	Forgy (F)	Picks *K* points at random as centers. Fast but unreliable.
Probabilistic	K-means++ (K)	Chooses centers with a probability based on distance. Improves quality.
Deterministic	Var-Part (V)	Splits data using variance. Fast and stable, good for large data.
Heuristic	Maximin (X)	Selects centers far apart. Simple but weak with complex data.

b) Convergence

The K-means algorithm is guaranteed to converge to a local minimum of the objective function—typically the Sum of Squared Errors (SSE)—because SSE decreases at each iteration and is bounded below by zero. This monotonic decrease ensures that the algorithm eventually stops. However, the specific local minimum reached, as well as the speed of convergence, can be heavily influenced by the choice of initialization and the underlying structure of the data. Methods like K-means++ improve convergence behavior by providing better initial cluster centers, often leading to higher-quality partitions and fewer iterations to converge [95].

c) Choice of number of Clusters

There is no universally optimal method for determining the best number of clusters in K-means, as it depends on the nature and distribution of the data. However, several heuristics and evaluation techniques are commonly used:

- **Elbow Method**: This intuitive approach involves plotting the total within-cluster sum of squares (SSE) against different values of *k*. The "elbow" point—where the rate of SSE decrease sharply slows—indicates a good balance between model complexity and explained variance. It is widely used due to its simplicity.
- **Silhouette Analysis**: This method evaluates how similar an object is to its own cluster compared to other clusters. The silhouette score ranges from -1 to 1, with higher values indicating better-defined clusters. The optimal number of clusters is typically the one that maximizes the average silhouette score.

5.2.3. Advantages and Limitations

The K-means algorithm is widely used due to its simplicity and efficiency, but it also comes with several limitations that must be considered when applying it to real-world data.

Table 2.3. Advantages and Limitations of the K-means Clustering Algorithm

Advantages	Limitations
Simple to understand and implement	Requires the number of clusters K to be specified in advance
Fast and computationally efficient	Sensitive to the initial placement of centroids
Scales well to large datasets	Assumes spherical, equally-sized clusters
Produces easily interpretable cluster centers	Not suitable for discovering non-convex or overlapping clusters
Widely supported and used in practice	Sensitive to noise and outliers
	May converge to local minima (not always global optimum)

6. Data Preprocessing and complementary techniques for Learning

To ensure effective training and accurate predictions in machine learning systems, proper data preprocessing and the integration of complementary techniques are essential. These steps aim to enhance data quality, reduce complexity, and improve model generalization

6.1. Dimensionality reduction

6.1.1. Definition and objectives

Dimensionality reduction is the process of transforming high-dimensional data into lower-dimensional data while preserving the original information and meanings of the data as much as possible. This process helps simplify complex data to facilitate its analysis and processing [96]. Its main objectives include:

- Reducing the number of variables or features without losing important information.

- Reducing the time and resources required for operation by decreasing the input dimensions to the system.
- Removing excessive, duplicate, or noisy data that may hinder the model's performance and accuracy.
- Eliminating the curse of dimensionality, which means reducing the negative effects resulting from high-dimensional data that may lead to overfitting.
- Facilitating the interpretation of patterns or relationships between data by presenting them in lower dimensions

6.1.2. Principal component analysis

Principal component analysis (PCA) aims to reduce the high dimensions of data by transforming it from a high-dimensional space to a low-dimensional space while preserving the maximum amount of information present in the original data.

PCA is a statistical analysis method that transforms the original data into a new space consisting of a limited number of principal components that are uncorrelated and orthogonal, representing the directions of maximum variance in the data.

This method is performed by determining a new set of orthogonal axes called principal components, which represent the directions containing the greatest variance in the data. Each data sample is projected onto this new space, resulting in a more compressed representation that preserves the underlying structure of the data.

a) Mathematical principal of PCA

Here are the steps to extract the principal components (A tutorial on Principal Components Analysis) :

- **Mean Centering :** Where we calculate the mean for each feature and subtract it from the values, i.e., we center the data around zero.
- **Covariance Matrix computation:** This matrix C illustrates how the data are interconnected, meaning how the variables affect each other. It is calculated using this formula:

$$C = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})\,(X_i - \bar{X})^T$$

  While X_i is the data sample and $\bar{X}$ is the mean
- **Eigen Decomposition:** it involves finding the eigenvectors and eigenvalues of the covariance matrix C. The eigenvectors represent the directions (principal components), while the eigenvalues indicate the amount of variance captured in each direction.
- **Selection of Principal Components**: by ordering the eigenvectors according to their eigenvalues and selecting the most significant ones. Thus, the top k eigenvectors are selected, where k is the number of dimensions to retain.

- **Re-representation of data:** or projection of data onto the newly selected axes, thereby reducing dimensions.

b) Application in images processing

In image processing, PCA is widely used as a dimensionality reduction technique to compress image data while preserving essential structural information. PCA is commonly applied in several tasks such as [97]:

1. Facial recognition, as it is used to extract the most important features to distinguish between facial images and reduce dimensions.
2. Classifying medical images, especially MRI images, by enhancing them through the removal of repetitive features.
3. Hyperspectral image processing where PCA allows for real-time handling of hyperspectral images by reducing dimensions.
4. Noise removal and preprocessing in ultrasound or X-ray images, such as detecting heart disorders.
5. Image compression to reduce storage size and slight quality loss without losing the most important components.

c) The relevance in the context of machine learning

Principal Component Analysis (PCA) is considered a very important preliminary step in machine learning before classification and clustering algorithms. It is used to reduce dimensions while preserving the main components, which helps in reducing computational complexity, increasing the effectiveness and accuracy of models, removing noise, and eliminating redundancy. This enhances machine learning algorithms and elevates their capabilities to a higher level, while reducing the risk of overfitting and making it easier to visualize high-dimensional data [97].

6.2. Other data preprocessing techniques

In addition to dimensionality reduction, several preprocessing techniques are essential to prepare data for effective learning and improve model performance:

6.2.1. Normalization and standardization

These techniques scale features to a common range or distribution.

Normalization is the process of transforming data into a specific range, such as between 1 and -1. It has a great importance and necessary when there are significant differences and distances between the data and the value ranges of different features [98].

Standardization, on the other hand, adjusts the data so that it has a mean of zero and a standard deviation of one. This technique is particularly appropriate when the features are assumed to follow a Gaussian distribution [99].

6.2.2. Data cleaning and handling missing values

To enhance the performance of machine learning models, it is essential to improve data quality, as poor data leads to inaccurate results. Data cleaning techniques address issues like corrupted, duplicated, or incomplete data. With advances in AI, smart tools like IBM Data Quality for AI have emerged to automatically improve dataset quality. Machine learning itself can now learn data cleaning rules without the need for manual rule writing [100].

Handling missing values is another critical aspect of data preprocessing, as such gaps in the dataset can introduce bias and degrade model performance. Missing data refers to the unrecorded values in a dataset, which can be of significant importance in the analysis process. These gaps often result from human errors, privacy issues, or technical malfunctions. Missing data negatively affects data quality, leading to decreased model accuracy and effectiveness. Therefore, several techniques are used to handle this type of data, such as regression, decision trees, and deep learning methods, with the aim of compensating for the missing values and improving model performance [101]

6.2.3. Data augmentation

Data augmentation is a technique aimed at improving the performance of machine learning models as a whole by expanding the size of the data and thus artificially increasing the training data. This is done by modifying the original data to obtain new and diverse data that helps reduce the problem of insufficient training data and improve the model's generalization. There are various methods for data augmentation, such as using advanced techniques like generative adversarial networks. This method is applied in many fields, including image processing, text, and tabular data [102].

7. Conclusion

This chapter has highlighted the fundamental role of machine learning and deep learning in solving complex computational tasks, particularly in the domain of image processing. With techniques such as supervised, unsupervised, and reinforcement learning, machine learning offers a diverse set of strategies tailored to various data types and learning objectives.

Convolutional neural networks (CNNs) have emerged as powerful tools, capable of automatically learning and extracting relevant features from images through hierarchical layers. Their effectiveness is further enhanced by the use of pre-trained models, which reduce training time and improve generalization. Additionally, ensemble learning techniques and preprocessing methods—such as data augmentation and dimensionality reduction—have proven essential in improving model performance and reducing error rates.

Altogether, the integration of these techniques within artificial intelligence systems provides a robust and flexible framework for addressing challenges in modern image analysis, supporting the development of solutions that meet both computational demands and real-world requirements.

Chapter 3. Conception

1. Introduction

In light of the increasing challenges related to processing deteriorated documentary images, traditional binary transformation techniques have become incapable of achieving effective, accurate, and reliable results—especially when there is an overlap between the background and the text, which makes it difficult to separate them using threshold-based methods. Consequently, deep learning emerges as a highly efficient option that overcomes the constraints of traditional methods through its ability to learn from previous examples and adapt to various types of degradation.

We have dedicated this chapter to presenting our deep learning-based methodology for performing the binarization process of deteriorated document images. In this chapter, we present a set of ideas aimed at providing a clear picture of the proposed methodology. The chapter covers the various stages involved in the process, along with a summary design of the approach, before concluding with a comprehensive summary.

2. Motivation

The main objective of this work is to develop an effective method for the binarization of deteriorated document images that suffer from various types of distortions such as stains, noise, and low contrast. Given the increasing demand for the preservation and analysis of cultural heritage, traditional binarization methods—such as global and local thresholding—often struggle to cope with the complexity and widespread defects found in degraded documents. These limitations have prevented such methods from achieving high levels of accuracy and efficiency.

This has led us to explore the field of artificial intelligence in search of more adaptive and robust solutions. After evaluating various strategies, we turned to deep learning—a powerful approach that enables models to learn complex representations directly from data and adapt to a wide range of degradation types.

Deep learning techniques allow for the automatic extraction of meaningful and discriminative features, enabling the system to better distinguish between text and background, even in cases of severe interference or noise. Furthermore, these techniques offer greater flexibility and generalization capabilities, making them suitable for diverse types of documents and degradation patterns.

Therefore, we believe that the adoption of deep learning represents a key step toward achieving accurate and reliable binarization of deteriorated document images.

3. Description of the proposed approach

Binarization can be approached from different perspectives. It may be viewed as a segmentation problem, where the goal is to separate text regions from the background. It can also be seen as a localization problem, where the task is to identify where text appears in the image. Most importantly, it can be formulated as a pixel-wise classification problem, where each pixel is classified as either foreground (text) or background. This is the perspective we adopt in our work.

Convolutional Neural Networks (CNNs) have shown excellent performance in image classification tasks, including pixel-level classification when properly adapted. However, a challenge arises from the fact that CNNs typically operate on image patches rather than individual pixels. To overcome this, we designed a patch-based strategy in which a pixel and its surrounding context are used as input to the CNN, allowing the network to infer the class of the central pixel based on local visual information.

The proposed method does not rely on thresholding techniques such as global or local thresholding, which are commonly used in traditional binarization approaches. Instead, it focuses on the direct classification of pixels as either text or background. This means that the method does not compute or learn any threshold value. Rather, it combines machine learning techniques—specifically K-means clustering and Principal Component Analysis (PCA) for dimensionality reduction—with Convolutional Neural Networks (CNNs) to convert grayscale images into binary (black-and-white) images.

The core idea behind this methodology is to train a CNN on a set of patches extracted from deteriorated document images, where each patch represents a small region of the image and is labeled based on the value of its central pixel, taken from the corresponding ground truth image. Once trained, the model is able to generalize and classify unseen images.

The proposed methodology consists of three main stages as it is described in Figure 3.1:

1. Training data preparation, where samples from text and background regions are extracted separately;
2. Model construction and training, where the CNN is built and trained on the labeled patches;
3. Testing and evaluation, where the trained model is applied to new images and its performance is assessed.

This methodology overcomes the strict limitations of traditional binarization methods by enabling the model to learn complex representations of textual structures and background patterns directly from the data.

3.1. Preparation of training data

The first step in our methodology involves the preparation of training data from a dataset of degraded document images. This data is used to train the Convolutional Neural Network (CNN) model.

CNNs are a form of supervised learning, meaning that each input must be associated with a corresponding output label. During training, the model's weights are updated at each iteration based on the difference between the predicted outputs and the actual target values. This difference, referred to as the error, is propagated backward through the layers of the network using the backpropagation algorithm. This algorithm relies on gradient computation to adjust the weights gradually, enabling the model to learn and improve its performance over time.

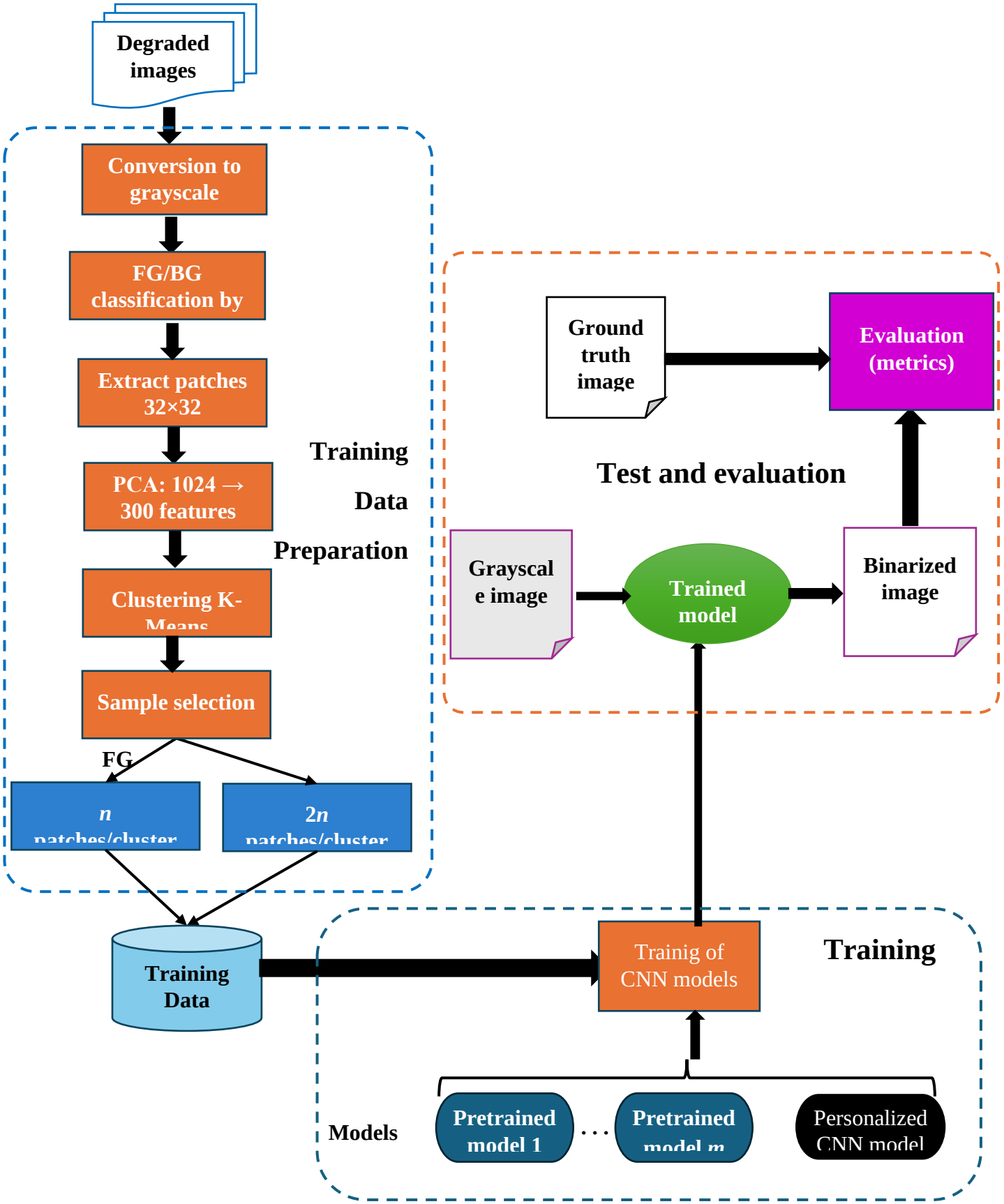
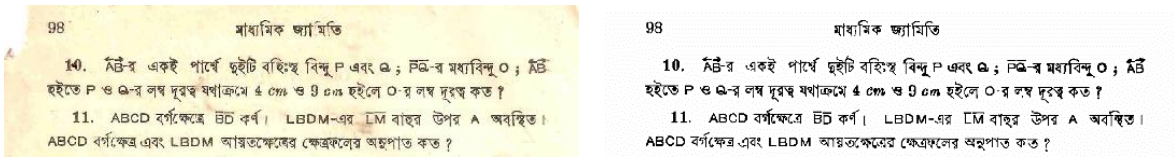


Figure 3.1. General Architecture of the Proposed Binarization System

In our project, the CNN learns to convert the pixels of degraded document images into binary images—comprising only black and white pixels—to clearly distinguish the text from the background. This binarization makes the documents easier to read and analyze.

To achieve this, we divided the degraded images into small patches, with each patch labeled as either text (foreground) or background. This patch-based strategy improves the model's ability to learn patterns related to the central pixel in each patch, increasing classification accuracy.

The ground truth images were preprocessed and aligned to ensure consistency with the input data. This step guarantees that each patch's label corresponds precisely to the class of its central pixel as derived from the ground truth. An example of a degraded input image and its corresponding ground truth is shown in Figure 3.2.



(a) **(b)**

Figure 3.2. Example of degraded document image and its ground truth, (a) Original image, (b) Ground truth image

This careful preparation of data—segmenting images into patches and assigning precise labels—allows the CNN to extract deep spatial features that are highly sensitive to subtle variations within the images. It also enhances the model's ability to generalize to various types of degradation and noise typically present in historical documents.

In our methodology, images are first converted to grayscale, then split into small patches. Each pixel is considered a central pixel, and its surrounding neighbors form a patch within a defined window of size $w \times w$. These patches serve as inputs to the CNN. The central pixel is then compared to the corresponding pixel in the ground truth image: if the pixel is white, the patch is labeled as background; otherwise, it is labeled as foreground. Examples of background and foreground patches are shown in Figure 3.3.

Before patch extraction, we apply PCA (Principal Component Analysis) to reduce dimensionality and K-means clustering to group pixels based on similarity. This step facilitates a more representative and diverse patch selection, providing the CNN with meaningful local information. As a result, the model becomes capable of distinguishing between text and background even in severely degraded images.

(a) **(b)**

Figure 3.3. Examples of extracted patches, (a) background patch, (b) foreground patch

3.1.1. Convert images into grayscale

As part of our methodology, especially in the data preparation phase for training, we rely on an essential initial step after loading the data: converting colored images to grayscale. This conversion plays a crucial role in the preparation process as it simplifies the visual information by removing color components without losing the light gradients necessary for distinguishing text from the background. Consequently, it reduces the complexity of the input data, allowing the model to focus primarily on differences in light intensity, which is sufficient for performing the task of distinguishing between text and background in document images.

Converting images to grayscale can be achieved using the following pseudocode.

Algorithm: Convert Color Image to Grayscale

Input: A color image

Output: A grayscale image

Begin

For each pixel *p* in the image:

r ← red channel intensity of *p*

g ← green channel intensity of *p*

b ← blue channel intensity of *p*

gray ← (*r* + *g* + *b*) / 3

Set the pixel *p* value to *gray*

End For

End.

This algorithm converts each pixel in the colored image into a single gray value by computing the average intensity of the red, green, and blue channels and assigning it as the gray pixel value.

The output of this algorithm is a grayscale image (Figure 3.3) used as a basis for extracting small patches surrounding each pixel in the image.

(a) **(b)**

Figure 3.4. Grayscale conversion, (a) Original Image, (b) its Corresponding Grayscale Version

This step is essential for a deeper understanding of the local context and analyzing the area surrounding the central pixel within each window. It also helps enhance the model's ability to

resist color changes resulting from the degradation of the original document's quality, thereby improving the accuracy of distinguishing between text and background.

3.1.2. Automatic segmentation of large images into sub-images

In our methodology, we encountered a significant challenge related to the processing of large-sized images. Images with dimensions exceeding 1000×1000 pixels contain millions of pixels, which, if handled directly, result in the extraction of an extremely large number of small patches. This causes issues in both storage and processing, leading to excessive memory consumption and longer computation times—especially in limited-resource environments.

To address this issue, we adopted a strategy that divides each original image, after converting it to grayscale, into smaller sub-images. This approach significantly reduces resource usage while maintaining processing efficiency.

The segmentation mechanism is based on the following conditions:

a) If the image is smaller than 1000×1000 pixels, it is processed as a whole without segmentation.

b) If the image is between 1000×1000 and 2000×2000 pixels, it is divided into 4 equal-sized sub-images.

c) If the image is larger than 2000×2000 pixels, it is divided into 9 sub-images arranged in a 3×3 grid.

These divisions help preserve image details while minimizing the memory and computational load. After segmentation, each sub-image is treated as an independent image from the document image processing dataset.

3.1.3. Feature Extraction and Training Sample Selection from Document Images

Document images, especially those that are degraded or historical, often exhibit a wide range of characteristics including uneven lighting, varying contrast levels, background noise, and structural inconsistencies in the text. Using every pixel in such images as a training sample may seem exhaustive, but it is neither efficient nor optimal. The vast number of pixels can introduce significant redundancy, as many neighboring regions share similar characteristics. Moreover, directly using all pixels would not only lead to a considerable computational and memory burden but could also result in an imbalanced dataset, with certain patterns or regions being overrepresented while others remain under-sampled.

To address this, our approach aims to extract a meaningful and diverse subset of training data that captures the essential variations within both the text (foreground) and background regions. Each document image is processed to extract local patches around pixels, preserving the spatial context necessary for learning. Dimensionality reduction using Principal Component Analysis (PCA) is then applied to retain the most informative features while reducing noise and redundancy. Following this, k-means clustering is used to group similar patches, and a representative set of samples is selected from each cluster. This process is conducted separately for foreground and background areas to maintain class balance and to ensure that the model is exposed to a wide variety of text and degradation patterns.

In this section, a detailed explanation of each stage will be provided.

a) Patch Extraction Using a Sliding Window Approach

As part of the data preparation process, we systematically extract patches after dividing large images into sub-images using a sliding window approach with a fixed size of 32×32 pixels. This window moves across the image with a stride of 1 pixel, ensuring that every pixel is treated as the center of a patch. This exhaustive coverage allows the model to capture subtle variations in the local neighborhood of each pixel.

Handling pixels near the image borders presents a challenge, as a full 32×32 window cannot be centered around edge pixels without extending beyond the image boundaries. To address this, we apply mirror padding, where pixel values near the borders are symmetrically reflected. This method helps preserve edge structures and avoids introducing artificial borders, unlike zero-padding.

For each extracted patch, the central pixel in the corresponding binary ground truth (GT) image is examined to assign a label. Specifically, let P be a patch centered on pixel p, and let $G(p)$ denote the value of p in the GT image. Then:

- If $G(p) = 0$, the patch is labeled as *foreground*.
- If $G(p) = 255$, the patch is labeled as *background*.

This labeling process helps the model associate local visual patterns with text or background classes. These local regions often contain critical features such as texture, edge orientation, or noise artifacts. The CNN leverages this spatial information to build a robust mapping from raw pixels to binary classes.

However, this exhaustive extraction process results in a very large number of overlapping patches—especially for high-resolution images—, which poses challenges in terms of storage, computational efficiency, and data redundancy. To address this, we apply dimensionality reduction (via PCA) and clustering (using k-means) to identify and retain only the most representative samples. This selective strategy enhances both the efficiency and effectiveness of the model training process.

b) Dimensionality reduction by PCA

Once the patches are extracted, each 32×32 patch is reshaped into a one-dimensional feature vector of size 1024, corresponding to the grayscale intensity values of the 1024 pixels within the patch. Given the large number of patches generated from the entire dataset, the resulting matrix of training samples becomes high-dimensional and computationally demanding to process.

To reduce the computational cost and memory requirements while retaining the most informative features, we apply Principal Component Analysis (PCA)—a standard technique for dimensionality reduction. PCA transforms the original high-dimensional vectors into a new subspace defined by the directions (principal components) along which the variance of the data is maximized.

In our implementation, we reduced the dimensionality of each patch vector from 1024 to 300 by retaining the top 300 principal components. This choice was made after analyzing the cumulative explained variance of the dataset. We observed that the first 300 components retained more than 95% of the total variance, meaning that the most informative aspects of the data were preserved while reducing noise and redundancy. This balance between dimensionality reduction and information retention makes the dataset more manageable and accelerates the subsequent steps, such as clustering and classification, without sacrificing performance.

The following algorithm summarizes the steps used to apply PCA in our methodology:

Algorithm: Dimentionality reduction

Input:

- D: A matrix of M vectors (each vector has 1024 features from a 32×32 patch).
- K: Number of principal components to retain (e.g., K = 300).

Output:

- D_PCA: Matrix of reduced vectors with K features per sample

Begin

1. Center the data:
   - Compute the mean vector μ of D
   - Subtract μ from each row of D to obtain the centered data matrix D_c
2. Compute the covariance matrix C:
   - $C = (D\_c^T \times D\_c) / (M - 1)$
3. Compute the eigenvalues and eigenvectors of C:
   - Find eigenvalues λ_i and corresponding eigenvectors v_i
4. Sort eigenvectors by descending eigenvalues:
   - Rank the eigenvectors v_i based on λ_i from highest to lowest
5. Select the top K eigenvectors:
   - Form projection matrix P using the first K eigenvectors
6. Project the data:
   - $D\_PCA = D\_c \times P$

Return D_PCA

End.

c) Clustering of reduced patches using K-Means

After reducing the dimensionality of each 32×32 patch into a compact 300-dimensional vector via Principal Component Analysis (PCA), we further organize the data through clustering. At this stage, the feature vectors are separated into two groups based on their class: foreground (text) and background. This distinction is crucial, as the statistical and structural characteristics of text pixels differ significantly from those of background pixels.

To capture the underlying structure of each class and reduce redundancy, we applied the K-means clustering algorithm independently to both categories. For each class, we set the number of clusters to K = 10, resulting in 10 clusters for foreground patches and 10 clusters for background patches.

After dimensionality reduction, each patch is represented as a feature vector of 300 dimensions. These vectors are then separated into two groups based on their ground truth classification: one for the foreground (text) class and the other for the background class. This separation allows for a more targeted analysis of the statistical and structural properties of each class.

This choice of K=10 was made to strike a balance between diversity and computational efficiency. A smaller number of clusters might fail to capture meaningful intra-class variation, while a larger number could introduce unnecessary complexity without significant performance gains.Thus, each class is ultimately represented by 10 cluster centers, which serve as key prototypes for the subsequent training phase.

The clustering process is formalized below. This algorithm is applied for each group (foreground and background separatelly.

Algorithm: K-Means Clustering of Feature Vectors

Input:

- D: A matrix of M reduced feature vectors (each of 300 dimensions), obtained after PCA.
- k: Number of clusters (e.g., k=10).

Output:

- C: A set of k cluster centroids.
- L: A list of length M, containing the cluster assignment label $L_i \in \{1,\ldots,k\}$ for each vector D_i .

Begin

1. Initialize: Randomly select k vectors from D as initial centroids $C=\{c_1,c_2,...,c_k\}$.
2. Repeat until convergence:
   - Assignment step:
     - For each vector D_i in D, compute the Euclidean distance to all centroids.
     - Assign D_i to the cluster whose centroid is closest.
   - Update step:
     - For each cluster $j \in \{1,...,k\}$, compute the new centroid c_j as the mean of all vectors assigned to cluster j.
3. Return:
   - The final centroids $C=\{c_1,...,c_k\}$
   - The cluster labels $L=\{L_1,...,L_M\}$.

End.

d) Selection of Representative Vectors from Clusters

Following the clustering step using the K-means algorithm, where the reduced feature vectors were grouped into $k = 10$ clusters for each class (foreground and background), the next step in our methodology involves selecting a subset of representative samples from each cluster to form the final training dataset.

Selecting representative samples is crucial to reduce redundancy while ensuring that the training data remains both compact and diverse. Rather than using all vectors assigned to each cluster—which would reintroduce computational and storage challenges—we identify a limited number of well-distributed samples that best characterize the internal structure of each cluster.

Although one might intuitively select the vectors closest to the cluster centers (as they represent the most typical examples), this approach may lead to redundancy. Vectors located near the center often exhibit highly similar characteristics, which can reduce the diversity of the training set and potentially limit the model's ability to generalize.

To address this, we adopt a random sampling strategy within each cluster. By selecting samples at random, we ensure a natural variation within the training set, avoid over-concentration around the cluster centers, and introduce sufficient variability to improve generalization. This method maintains simplicity, avoids the risk of selecting outliers, and ensures that each cluster is equally likely to contribute diverse examples.

This sampling process is applied independently for each of the 10 foreground clusters and the 10 background clusters.

In addition, we account for the natural class imbalance found in historical document images, where background pixels typically far outnumber foreground (text) pixels. To reflect this imbalance in our training set, we apply an asymmetric sampling strategy:

- From each foreground cluster, we select n representative vectors.
- From each background cluster, we select $2n$ representative vectors.

This combined strategy—random intra-cluster sampling and class-level balancing—ensures that the resulting training set captures the internal diversity of each class while avoiding redundancy. It enhances the robustness of the classifier by providing a realistic and varied training distribution, which is particularly important when dealing with degraded and diverse document images.

The resulting curated dataset provides the CNN with a focused, informative, and diverse subset of training data, improving learning efficiency and reducing the risk of overfitting caused by noisy or repetitive samples.

3.1.4. Final Preparation and Organization of Training Samples

After selecting the appropriate representative samples from each cluster, with controlled quantities for both the foreground and background classes, the final step in the data preparation process involves organizing and storing these vectors in a structured and reusable format. This organization is essential for enabling an efficient training pipeline and ensuring reproducibility.

To facilitate class separation and prevent confusion during model training, the selected vectors are stored in distinct files according to their class:

- **Foreground samples** (text pixels, corresponding to black pixels in the ground truth) are saved in one file.
- **Background samples** (non-text pixels, corresponding to white pixels in the ground truth) are saved in another.

Each file is named according to the original image from which the samples were derived. This naming convention allows for straightforward traceability and association between the extracted samples and their source documents. It also facilitates organized storage, easy indexing, and future operations such as balanced loading, augmentation, or per-image performance evaluation.

Each file is named based on the original image from which the samples were extracted. This naming strategy ensures clear traceability, simplifies storage management, and facilitates various post-processing tasks such as balanced sampling, dataset augmentation, or per-document performance evaluation.

This organization—by class and by source image—enhances the modularity and flexibility of the training dataset. It allows for efficient handling during training, validation, or testing, and ensures that the final dataset remains both compact and diverse while aligned with the structural requirements of the classifier.

3.2. Model definition and training

This part of the methodology focuses on the development and training of the classification model used to distinguish between text (foreground) and non-text (background) pixels in degraded historical document images. Following the careful extraction, reduction, and selection of representative samples described in Section 3.1, we now turn our attention to defining an appropriate learning model capable of leveraging these features to perform accurate classification. The goal is to train a robust classifier that can generalizes well to unseen degradations and maintains high performance across diverse document types.

To identify the most suitable classifier, we initially considered a wide range of classical and modern machine learning techniques. While classical models can be efficient in certain cases, they typically rely on handcrafted features and often struggle to model complex and non-linear decision boundaries, especially in scenarios involving high variability and noise, as found in degraded document images.

To overcome these limitations, we turned to deep learning-based models, which have shown superior performance in image analysis tasks. Among these, Convolutional Neural Networks (CNNs) have emerged as the most appropriate solution for our context. CNNs are particularly well-suited for document image analysis due to their localized feature extraction capabilities, allowing them to capture structural patterns and spatial dependencies within image patches. Their architecture—comprising convolutional layers, pooling operations, and fully connected layers—enables them to perform efficient filtering, dimensionality reduction, and final classification in a single, end-to-end trainable pipeline. Moreover, CNNs benefit from the

spatial neighborhood of each pixel, which is especially important in distinguishing between text and background regions in noisy or degraded settings.

Another advantage of CNNs is their flexibility: the architecture can be tailored in terms of depth, number of filters, kernel sizes, and activation functions to match the characteristics of the input data and the classification task. When trained on the structured and diverse dataset we prepared, the CNN exhibited a strong capacity to generalize across various forms of degradation and layout irregularities.

3.2.1. Model Definition

To identify the most effective architecture for pixel-level classification in degraded historical documents, we explored two complementary strategies: evaluating existing pretrained CNN architectures through transfer learning, and designing a custom CNN model tailored to our specific data and classification task.

a) Experimentation of pretrained models

Before developing a custom architecture, we first evaluated the potential of using existing pretrained convolutional neural networks that have been successfully applied in a wide range of computer vision tasks. These models, typically trained on large-scale datasets such as ImageNet, offer rich feature representations that can be transferred and adapted to new tasks through fine-tuning.

The rationale behind this step was to benefit from the robust and generalizable features learned by deep networks trained on massive and diverse image corpora, and to assess whether these features could be leveraged effectively in the context of degraded document classification. Although our task involves relatively small patches (e.g., 32×32 pixels) rather than full-sized natural images, pretrained models can still be valuable if properly adapted and fine-tuned on domain-specific data.

The models we experimented with are:

- **Modernised LeNet**: LeNet was originally proposed for digit recognition and is specifically designed to work with small 32×32 grayscale images, which aligns perfectly with the size of our extracted patches. Its simplicity and efficiency made it a natural first choice for benchmarking. We used a modernised version that includes batch normalization and dropout to reduce overfitting.
- R**esNet**: ResNet was selected because of its deep architecture with residual connections, which helps overcome the vanishing gradient problem and allows for learning richer features. It is particularly known for its ability to generalize across complex visual patterns.
- **MobileNetV2:** This lightweight and efficient model, designed for mobile and embedded applications, uses depthwise separable convolutions to reduce computation. It supports transfer learning and can be adapted easily for custom classification tasks.

To adapt these models to our task, we modified the input layers to handle 32×32 grayscale patches and replaced the final classification layers with new fully connected layers tailored to our binary classification objective (foreground vs. background). The models were then fine-tuned using our curated dataset of representative text and background samples.

The results showed that the modernised LeNet performed well and demonstrated good generalization for distinguishing between foreground and background classes. ResNet, although capable of learning deeper patterns, exhibited overfitting due to the relatively small and specific nature of our dataset. MobileNetV2 achieved the best performance overall, balancing accuracy and speed effectively, and benefiting from transfer learning and its lightweight architecture.

These findings motivated us to design a custom CNN architecture, more lightweight and tailored to our particular classification problem. This custom model is described in the following subsection.

b) Design of the proposed CNN model

After experimenting with various pretrained models, we decided to develop a custom convolutional neural network specifically tailored for pixel-level classification in degraded historical document images. The objective was to construct a lightweight yet effective model capable of classifying 32×32 grayscale patches as either foreground (text) or background (non-text), while minimizing overfitting and maintaining computational efficiency.

The decision to build a dedicated model was driven by several observations. First, while pretrained models like ResNet and MobileNetV2 are powerful, they were originally developed for large-scale natural image classification and often include a large number of parameters that are unnecessary for our binary classification task. Second, our dataset is composed of small, well-structured image patches, which allowed us to adopt a more compact architecture without sacrificing accuracy. Third, a custom design gave us full control over the number of layers, filters, and kernel sizes, enabling us to fine-tune the architecture to better suit the spatial and statistical properties of historical document degradations.

The proposed model was designed with a focus on balancing architectural depth with computational cost, ensuring the extraction of distinctive features from fine-grained image patches. It follows a sequential architecture composed of layers carefully arranged to harmonize model complexity, feature learning capacity, and regularization.

As illustrated in Figure 3.5, the architecture consists of two primary convolutional blocks. Each block begins with a Conv2D layer that extracts initial local features, followed by Batch Normalization to stabilize activations and accelerate convergence. A second Conv2D layer further deepens the feature representation, again followed by Batch Normalization. Spatial resolution is then reduced via a MaxPooling2D layer, which also lowers the computational load. A Dropout layer is included afterward to reduce the risk of overfitting by randomly disabling neurons during training.

Following the convolutional blocks, the output is flattened into a one-dimensional vector using a Flatten layer. This vector is fed into a Dense layer, followed by an additional Dropout layer to further enhance generalization. Finally, a second Dense layer with a sigmoid activation function performs the binary classification into foreground and background categories.

This design combines hierarchical learning of visual features with modern regularisation techniques to achieve accurate and stable performance and reduce the risk of overfitting.

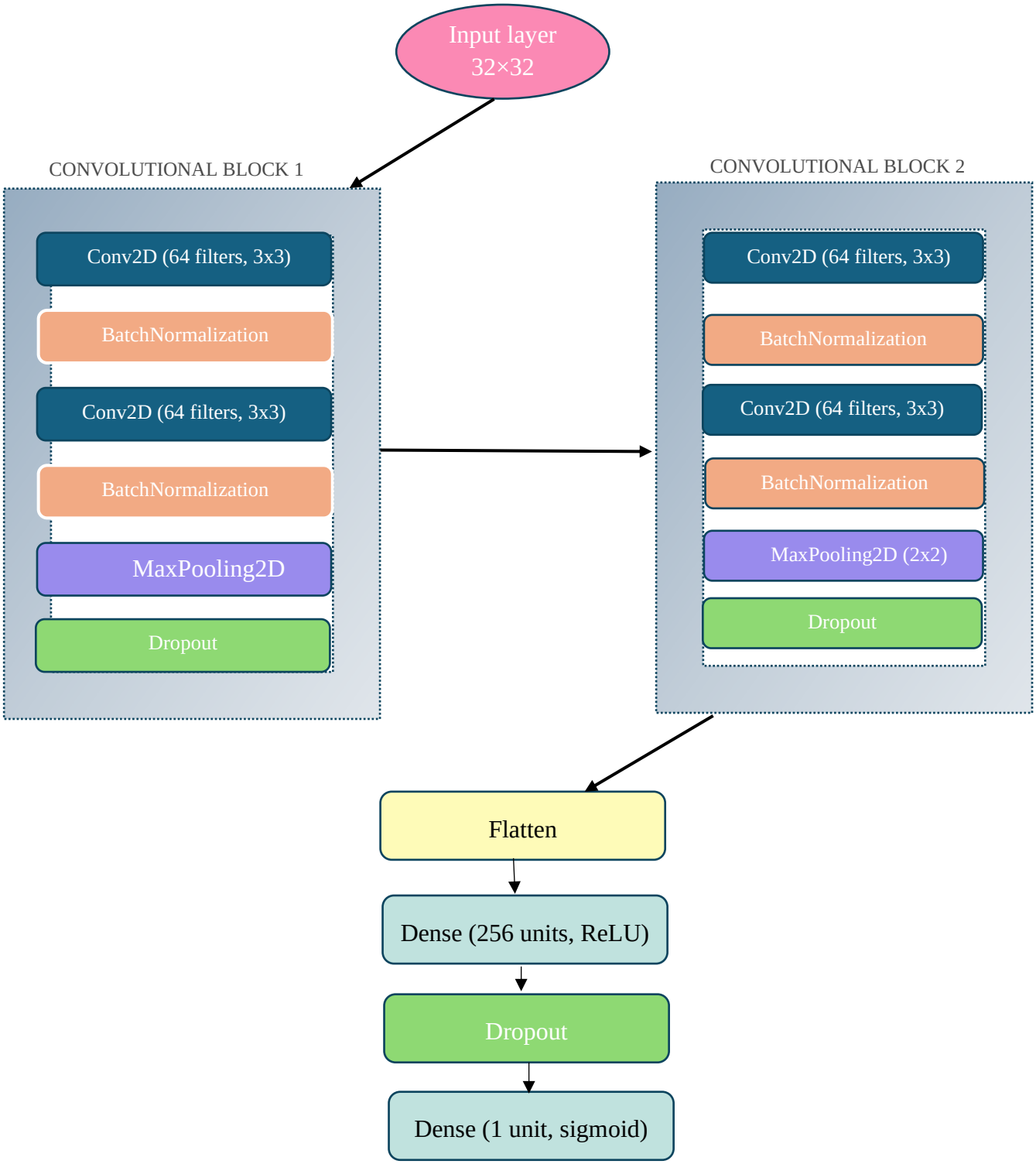


Figure 3.5. Overview of the Proposed Model Architecture

3.2.2. Model Training

Once the model architecture was defined—whether using a pretrained network or a custom-designed CNN—the next phase involved training the model to perform binary classification of document image patches as foreground (text) or background (non-text). This training process is framed within a supervised learning paradigm, where each input patch is associated with a ground truth label derived from annotated binary masks.

The objective of training is to optimize the model's parameters so that it can learn meaningful representations of the visual and structural differences between the two classes. Through successive updates of its internal weights, the model learns to associate specific spatial and statistical patterns with either the foreground or the background class.

To ensure a reliable and effective training process, the dataset was divided into two subsets: 80% of the samples were used for training and 20% were reserved for validation. This separation enabled ongoing monitoring of the model's generalization ability throughout the training process and helped prevent overfitting.

The design of the training process incorporated well-established regularization strategies, such as dropout and batch normalization, which were already integrated within the model architectures. These techniques are especially crucial in the context of degraded document images, where variations in noise, background clutter, and ink degradation can introduce instability into the learning process if not properly mitigated.

Moreover, to help the model generalize across different document types and degradation conditions, the curated training dataset (as described in Section 3.1) included a balanced and diverse collection of samples. This diversity ensures that the model is exposed to a wide range of structural configurations and degradation patterns, enhancing its robustness and adaptability when applied to unseen documents.

The training process itself can be summarized by the following pseudocode:

Algorithm: model training

Input: Labeled dataset of image patches {*X*, *Y*}

Output: Trained model CNN* (best performing on the validation set)

Begin

- Split the dataset into training set (*X_train*, *Y_train*) and validation set (*X_val*, *Y_val*)
- Initialize CNN model with the defined architecture
- For each epoch:
  - For each batch in X_train:
    - Forward propagate inputs through the network
    - Compute loss between predictions and ground truth labels
    - Backpropagate errors and update model weights
  - End for
  - Evaluate model performance on *X_val*
  - Apply regularization (e.g., dropout, batch normalization)
  - Monitor training and validation metrics (e.g., accuracy, loss)
  - Apply early stopping or adjust learning rate if needed
- End for
- Return the trained model CNN*

End.

3.3. Application of the Trained Model for Document Binarization

This stage constitutes the final phase of the proposed methodology, where the trained model is deployed to perform binarization on degraded historical document images. The objective here is twofold: first, to apply the model in a real-world scenario, and second, to rigorously evaluate its generalization performance on a separate test dataset that was completely excluded from the training and validation phases.

The prediction process involves systematically dividing each test image into overlapping patches of size 32×32, in alignment with the configuration used during training. For each pixel, a patch centered around it is extracted and normalized before being fed into the trained CNN model. The model then produces a binary prediction indicating whether the central pixel of the patch belongs to the foreground (text) or the background (non-text).

This patch-wise classification is repeated across the entire image, resulting in a complete binary mask that highlights the text regions while suppressing background noise (figure 3.6). This fully automated inference process leverages the spatial and contextual knowledge the model has acquired during training, making it particularly effective even in the presence of severe degradations such as ink bleed-through, stains, or faded text.

The resulting binarized images can then be subjected to further qualitative and quantitative evaluations to assess the model's robustness, accuracy, and suitability for downstream tasks such as OCR or digital archiving.

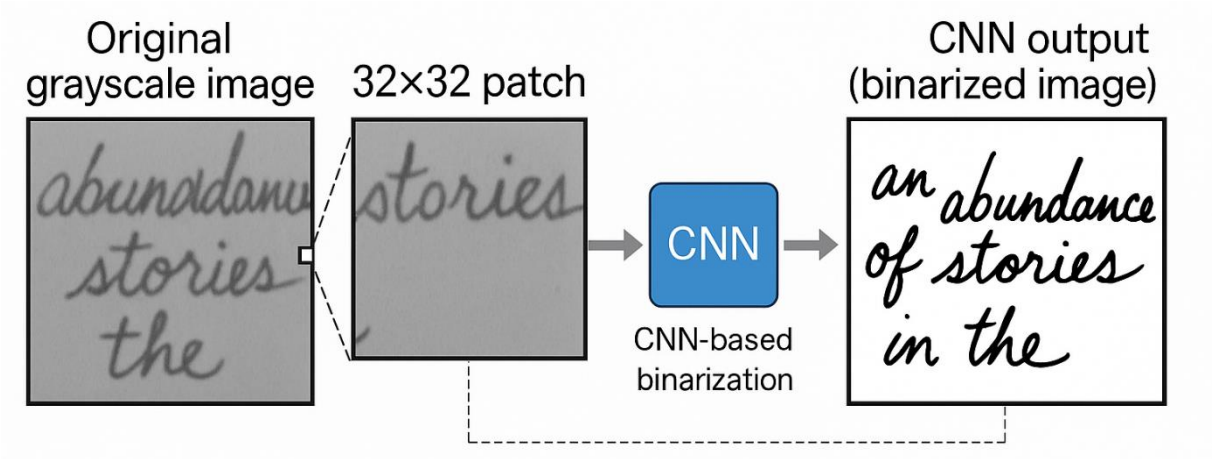


Figure 3.6. System Workflow Representation for Document Image Binarization

4. Conclusion

In this chapter, we presented a comprehensive methodology for designing a robust system for the binarization of degraded historical document images. The proposed approach is structured around three essential components: data preparation, model definition and training, and the application of the trained model for binarization.

The data preparation process involved extracting representative image patches and applying dimensionality reduction and clustering techniques to ensure compact, informative, and balanced training sets. We then explored several deep learning strategies, including the evaluation of pretrained convolutional neural networks and the design of a custom CNN

architecture tailored to the characteristics of document degradation. The model training process was guided by best practices to ensure generalization, including regularization techniques and a carefully curated dataset.

Finally, the trained model was integrated into a binarization pipeline capable of processing full document images by classifying image patches at the pixel level. This pipeline forms the core of the proposed binarization system, designed to perform reliably across diverse degradation types and document structures.

The details of the implementation, including model configuration, training parameters, and the complete experimental setup, are presented in the next chapter.

Chapter 4. Implementation and Results

1. Introduction

After completing a comprehensive presentation of the topic and defining the methodology with precision and clarity, we now proceed in this chapter to present the system implementation process in detail and provide the results obtained. We begin with an overview of the adopted development environment, including the programming language and libraries used, while highlighting the system's key components and functionalities. This is followed by a presentation of the experimental results, accompanied by an analysis of the main performance indicators and a discussion of their implications. Finally, a practical demonstration of the application is provided to illustrate how the system performs in real-world scenarios.

2. Development environment

In this section we review the programming language, libraries, and tools used in implementing the project and building the system.

2.1. Programming language

Python was used in the implementation of this project due to its simplicity and effectiveness in the field of image processing and deep learning, especially when working with deep models. It is also widely adopted thanks to its rich ecosystem of specialized libraries such as OpenCV and Keras.

2.2. Kaggle

The Kaggle platform is a comprehensive environment for developers and data scientists, allowing them to upload real-world datasets and develop predictive models using programming languages such as Python and R.

Kaggle offers a cloud-based programming environment that supports Jupyter Notebooks, with ready-to-use data science libraries and the ability to customize the work environment. Participants are evaluated based on their model performance using precise metrics, and their results are displayed in public and private leaderboards.

Thus, Kaggle represents a powerful educational and competitive platform that combines practical learning with real-world challenges [103].

2.3. GPU Acceleration

The Kaggle platform provides access to free Graphics Processing Units (GPUs), which significantly accelerate complex computations, especially in the fields of deep learning and image processing. Among the available GPUs is the T4 x2 GPU, which can be activated within the Kaggle Notebooks environment.

These GPUs notably enhance the speed of executing deep learning models and facilitate efficient training processes [104].

2.4. Libraries

In this section we represent the set of libraries used in our project :

2.4.1. TensorFlow

It is an open-source framework developed by Google for the creation of machine learning and deep learning models. It utilizes a dataflow architecture that depicts processes as graphs of "tensors" capable of being handled by CPUs, GPUs, and TPUs. It provides users with high-level and low-level programming interfaces to facilitate development and experimentation, and it is commonly used in cloud-based artificial intelligence applications [105].

2.4.2. Numpy

Numpy represents the core package for scientific computing in Python and it is a library that provides multi-dimensional array objects. Additionally, it offers derived objects such as masked arrays and matrices, along with a set of functions that enable fast and efficient mathematical, logical, shape, sorting, and selection operations. In our project, Numpy helped in representing and processing data quickly and efficiently, especially in handling arrays and converting images into formats suitable for models [106].

2.4.3. OpenCV and PIL

OpenCV is considered an open-source library used in the field of computer vision, providing more than 2500 efficient algorithms that cover various tasks such as face detection, object tracking, scene classification, image stitching, and 3D model extraction, making it a powerful tool in artificial intelligence and image processing applications [107].

As for PIL, its extension is a Python library for image processing, currently known as Pillow. It provides users with extensive support for image formats and interfaces to access pixel data, in addition to tools for image processing such as cropping, rotating, and so on, making it play a significant role in image processing [108].

2.4.4. Keras

Keras is considered a high-level API for developing deep learning models written in Python and can run on top of TensorFlow, JAX, or PyTorch. It is characterized by simplicity and ease of use, thereby reducing the burden on developers. It is also known for its flexibility and scalability, making it suitable for both simple and complex models [109].

2.4.5. Sickit-Learn

This library is considered one of the most famous open-source libraries for machine learning in Python and provides effective tools for supervised and unsupervised learning, along with preprocessing techniques and model evaluation [110].

We used this library in our project to reduce dimensions using the PCA algorithm and to apply the K-means algorithm to cluster the vectors representing the pixels, as well as in the context of mixing samples to improve representation quality.

2.4.6. Matplotlib

The Matplotlib library represents a comprehensive tool for creating static, animated, and interactive plots in Python. We used it in our project to visualize image data and intermediate results, such as displaying patches or comparing original and final results [111].

2.4.7. Random

The random library is integrated into Python and is used to generate random numbers or select random elements from a sequence [112]. In our project, it was used to select random representatives from each cluster after applying the K-means algorithm.

2.4.8. Doxapy

It is a library specialized in converting images to binary format using local adaptive thresholding algorithms [113].

In our project, we used the library during the model evaluation phase, where it allowed us to calculate precise benchmark indicators such as DRD and NRM.

2.4.9. Sickit-image

scikit-image is an open-source library that includes a set of algorithms for image processing. Its features include high quality and an active community [114]. It was used in our project to calculate the PSNR index and apply the skeletonize algorithm to extract the skeleton of texts in images.

3. Experiments and results

In this section, we present the experiments conducted as part of the noise classification project for historical document images. These experiments aimed to evaluate the effectiveness of various deep learning architectures in recognizing different noise patterns, along with a rigorous assessment of their performance in realistic conditions.

The experimental setup was carefully constructed, relying on a specialized dataset composed of historical document images. This dataset was divided into training, validation, and testing subsets to ensure a robust and unbiased evaluation of the models. Performance was measured using standard metrics such as accuracy, F1-score, PSNR, and others.

We begin by providing a summary of the dataset used and its structure. Subsequently, we present and analyze the results obtained from different models, including both individual architectures and combined configurations.

3.1. Datasets used

3.1.1. Source Datasets

The dataset used for training and validation was derived from a collection of well-known benchmark datasets commonly used in document image binarization research. These include both DIBCO and non-DIBCO datasets, which provide binary ground truth images for supervised learning.

As summarized in Table 4.1, the following 15 datasets were used to construct the training and validation patches:

Table 4.1. Summary of Document Image Datasets Used

Dataset	Number of Images
DIBCO 2009 [50]	10
H-DIBCO 2010 [51]	10
DIBCO 2011 [52]	16
H-DIBCO 2012 [53]	14
DIBCO 2013 [54]	16
H-DIBCO 2014 [55]	10
H-DIBCO 2016 [56]	10
DIBCO 2017 [57]	20
H-DIBCO 2018 [58]	10
DIBCO 2019 [59]	20
PHIBD [60]	15
CMATERdb 6 [61]	5
Bickley Diary [62]	7
ISOS Bleed-Through [63]	50
Nabuco [115]	35
Total	**248**

However, only the H-DIBCO 2016 dataset was excluded from training and validation and was reserved exclusively for blind testing. This decision ensures that our model is evaluated on truly unseen data for a realistic assessment of its generalization ability.

3.1.2. Constructed Patch-Based Dataset

Using the data preparation methodology described in Chapter 3, including preprocessing, dimensionality reduction via PCA, and clustering with K-Means, we generated a large-scale dataset of image patches. Each patch is a 32×32 pixel region, labeled according to its classification in the ground truth image as either foreground or background.

As shown in Table 4.2, the final patch dataset consists of 712,732 patches. The patches were randomly split into 80% for training and 20% for validation. Additionally, the 10 full images from H-DIBCO 2016 were used as an independent blind test set, to evaluate the system's ability to binarize full document images that were never seen during training.

Table 4.2. Distribution of patches in the final dataset.

Subset	Percentage	Number of Patches	Description
Training	80%	570,185	Used to train the model
Validation	20%	142,547	Used to monitor performance during training
Testing	—	10 full images (from H-DIBCO 2016)	Used to evaluate binarization performance
Total	**100%**	**712,732**	

3.2. Evaluation measurements

To comprehensively assess the performance of the proposed system and its alternative variants, we relied on a diverse set of objective evaluation measurements that capture different aspects of both classification accuracy and binarization quality. Accordingly, two distinct sets of metrics were employed: the first set focuses on evaluating the patch-level classification performance of the models, while the second set targets the assessment of binarization quality at the level of entire document images.

3.2.1. Classification metrics

The following standard metrics were used to evaluate the model performances:

- **Confusion matrix :** A square matrix used to evaluate the performance of a specific classification model, where each cell contains a positive integer representing the number of samples classified within each correct and incorrect prediction.[116]
- **Accuracy :** Accuracy is a metric used to evaluate the performance of a model in classifying data, and it reflects its ability to correctly and effectively distinguish between positive and negative classes. This metric is calculated as the ratio between the number of correct predictions, meaning those classified correctly as positive or negative, and the total number of all cases [117]. we can use this formula to calculate accuracy :

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

- **Precision :** Precision is calculated as the ratio of the number of pixels correctly classified as text (true positives) to the total number of pixels classified as text (true positives plus false positives) [118].

$$pecision = \frac{TP}{TP + FP}$$

- **Recall :** It is a metric that reflects the model's capability and performance, calculated as the ratio of the number of correctly classified pixels to the total number of true pixels in the input [118].

$$\text{recall} = \frac{TP}{TP + FN}$$

- **F1-Score:** The harmonic mean of precision and recall, providing a balanced measure that accounts for both false positives and false negatives.

3.2.2. Image-Level Binarization Metrics

To evaluate the quality of the binarized images produced by the models, we used a set of well-established metrics from the document image analysis literature. These metrics were selected for their ability to measure various aspects of binarization performance, such as preservation of textual detail, contrast enhancement, and noise suppression. As introduced in Chapter 1, Section 6, the following metrics were employed:

- F-Measure
- Pseudo-F-Measure

- Peak Signal-to-Noise Ratio (PSNR)
- Negative Rate Metric (NRM)
- Misclassification Penalty Metric (MPM)
- Distance Reciprocal Distortion (DRD)

3.3. Hyperparameter Configuration

The performance and generalization ability of a deep learning model are highly influenced by the choice of hyperparameters. In this work, the model was trained using a combination of core and advanced hyperparameters to ensure stable convergence, effective regularization, and optimal learning dynamics. Beyond the standard settings such as the optimizer, learning rate, and number of epochs, additional techniques like early stopping, learning rate scheduling, and dropout regularization were employed. These elements collectively contributed to controlling overfitting, accelerating convergence, and improving binary classification accuracy in the task of degraded document image binarization.

Table 4.3. Summary of the hyperparameter configuration used for training the proposed model.

Hyperparameter	Value	Description / Purpose
Optimizer	Adam	Adaptive optimizer suitable for CNNs
Learning rate	0.0005	Chosen for stable and smooth convergence
Batch size	64	Balances training speed and memory usage
Epochs (max)	200	Defines the upper training limit
EarlyStopping	patience=5, restore_best_weights=True	Stops training when validation loss stops improving
Loss function	BinaryCrossentropy	Appropriate for binary classification
Activation functions	ReLU (hidden), sigmoid (output)	ReLU speeds up training; sigmoid outputs a probability
Dropout rate	0.3–0.5	Prevents overfitting by randomly deactivating neurons
Learning Rate Scheduler	ReduceLROnPlateau(factor=0.5, patience=3)	Dynamically reduces learning rate if validation loss stagnates
Validation split	0.2	Reserves 20% of data for validation
Shuffle during training	True	Ensures samples are randomly mixed each epoch
Metrics	accuracy, F-measure, pseudo F-measure, PSNR, DRD, and NRM	Evaluate both classification and visual quality

3.4. Experiment 1: Searching for the best architecture

In this section, we present the experiments conducted to identify the most suitable CNN architecture for classifying text (foreground) and non-text (background) pixels in degraded historical document images. We began by adapting and evaluating three pretrained architectures—Modernized LeNet, ResNet20, and MobileNetV2—to the task. Following the

analysis of their performance, we proposed a lightweight custom CNN tailored specifically to the binary classification problem at hand.

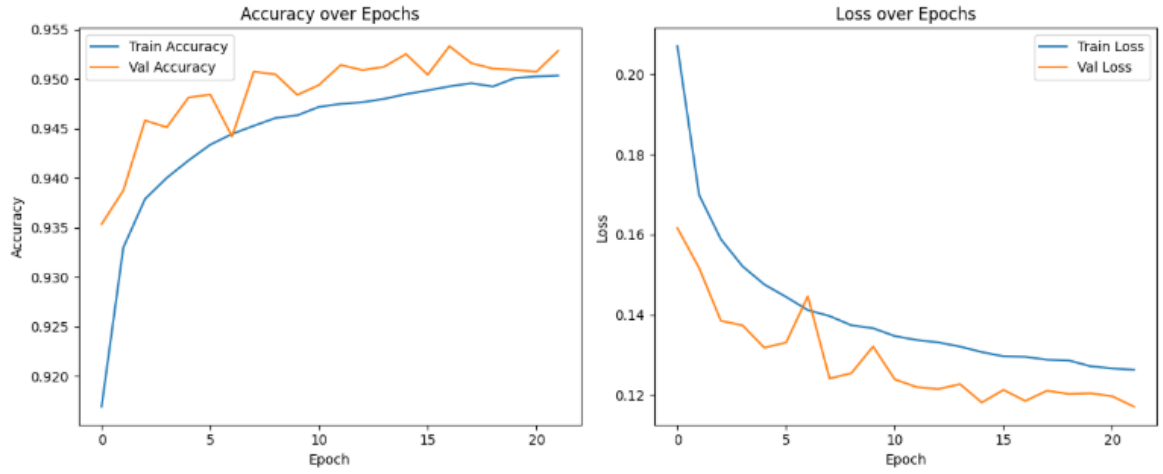
Each model is evaluated using the same dataset split, and performance is measured using standard metrics: accuracy, precision, recall, and F1-score. The results are presented and discussed separately for each model, under two main aspects: training behaviour and evaluation results on the validation set using a confusion matrix.

3.4.1. Modernized LeNet

To begin, we adapted the classic LeNet architecture to better suit our task by introducing modern techniques such as batch normalization and dropout. The model was trained on 32×32 grayscale image patches extracted from degraded document images.

a) Training Behaviour

Figure 4.1 below shows the model's training and validation curves over the training epochs:



(a) *(b)*

Figure 4.1. Modernized LeNet Training Progress: (a) training and validation accuracy, (b) training and validation loss

Figure 4.1.a shows that training accuracy steadily improves from 90% to 95%, with validation accuracy closely following, reaching 95.33% at the final epoch. This indicates consistent generalization throughout training.

Figure 4.1b displays the loss curves. The training loss drops from 0.24 to 0.12, while the validation loss decreases from 0.16 to 0.11. The parallel improvement of both curves reflects a stable learning process with minimal overfitting.

b) Confusion Matrix and Performance Metrics

After training, we evaluated the model using the validation set. The confusion matrix and performance metrics are presented in Tables 4.4 and 4.5:

Table 4.4. Confusion matrix for Modernized LeNet

Actual / Predicted	Background	Foreground	Total
Background	91,241	3,951	95,192
Foreground	2,702	44,653	47,355
Total	93,943	48,604	142,547

Table 4.5. Classification performance for Modernized LeNet

Metric	Background	Foreground	Average
Precision	0.9712	0.9187	0.9450
Recall	0.9585	0.9429	0.9507
F1-Score	0.9648	0.9307	0.9477

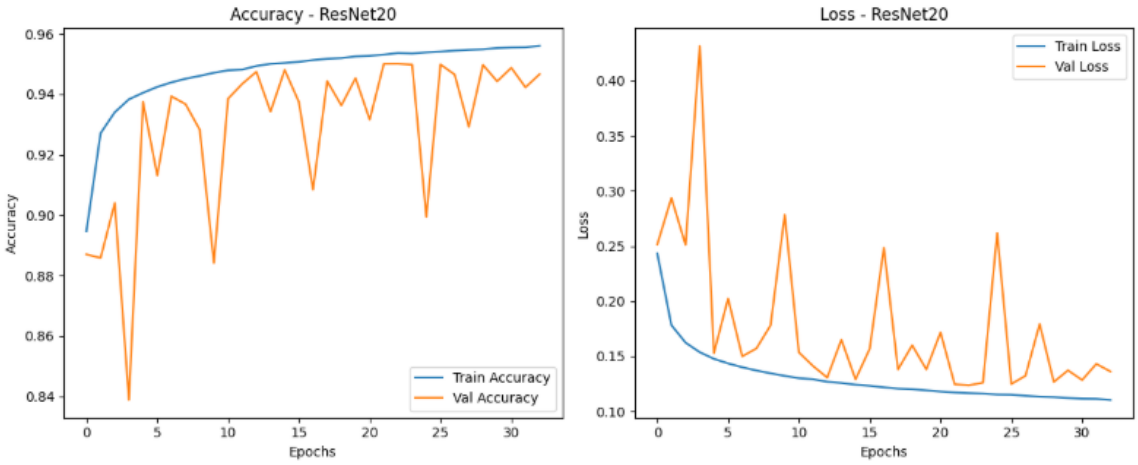
The model demonstrates solid performance, especially in recognizing background pixels. However, the slightly lower precision for foreground indicates occasional misclassification of text regions.

3.4.2. ResNet20

Next, we adapted a ResNet20 architecture for this task. This network is deeper and includes residual connections to facilitate the training of more complex representations.

a) Training Behaviour

Figure 4.2 shows the training behaviour of ResNet20:



(a) *(b)*

Figure 4.2. ResNet20 Training Progress: (a) training and validation accuracy, (b) training and validation loss

Figure 4.2.a shows training accuracy increasing steadily, with validation accuracy stabilizing around 95%. The high accuracy across both sets suggests the model effectively learns useful features. Figure 4.2.b shows a consistent decline in training loss. Although the validation loss remains low, some fluctuations are observed, suggesting the model may be sensitive to specific validation samples or slightly overfitting.

b) Confusion Matrix and Performance Metrics

Table 4.6. Confusion matrix for ResNet20

Actual / Predicted	Background	Foreground	Total
Background	91,241	3,951	95,192
Foreground	2,702	44,653	47,355
Total	93,943	48,604	142,547

Table 4.7. Classification performance for ResNet20

Metric	Background	Foreground	Average
Precision	0.9712	0.9187	0.9450
Recall	0.9585	0.9429	0.9507
F1-Score	0.9648	0.9307	0.9477

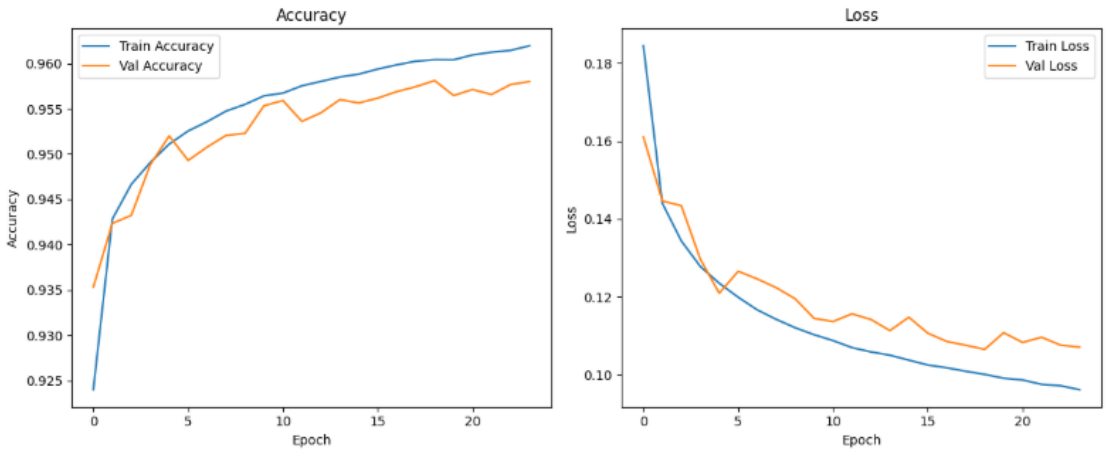
Performance is similar to LeNet. Although ResNet's deeper structure allows it to learn richer features, the results do not significantly surpass those of the simpler model and the network showed some sensitivity to overfitting.

3.4.3. MobileNetV2

MobileNetV2 is a lightweight architecture optimized for efficient computation. We adapted it for grayscale patches and modified the top layers for binary classification.

a) Training Behaviour

Figure 4.3 illustrates MobileNetV2's training process:



(a) *(b)*

Figure 4.3. MobileNetV2 Training Progress: (a) training and validation accuracy, (b) training and validation loss

Figure 4.3.a shows that training accuracy initially reaches 96% but decreases gradually to around 92.5%, indicating a slight degradation in performance as training progresses. Figure 4.3.b shows a progressive decline in both training and validation loss, converging at around 0.10. Despite the accuracy drop, the convergence of the losses suggests improved stability and generalization.

b) Confusion Matrix and Performance Metrics

Table 4.8. Confusion matrix for MobileNetV2

Actual / Predicted	Background	Foreground	Total
Background	92,085	3,107	95,192
Foreground	2,866	44,489	47,355
Total	94,951	47,596	142,547

Table 4.9. Classification performance for MobileNetV2

Metric	Background	Foreground	Average
Precision	0.9698	0.9347	0.9523
Recall	0.9674	0.9395	0.9534
F1-Score	0.9686	0.9371	0.9528

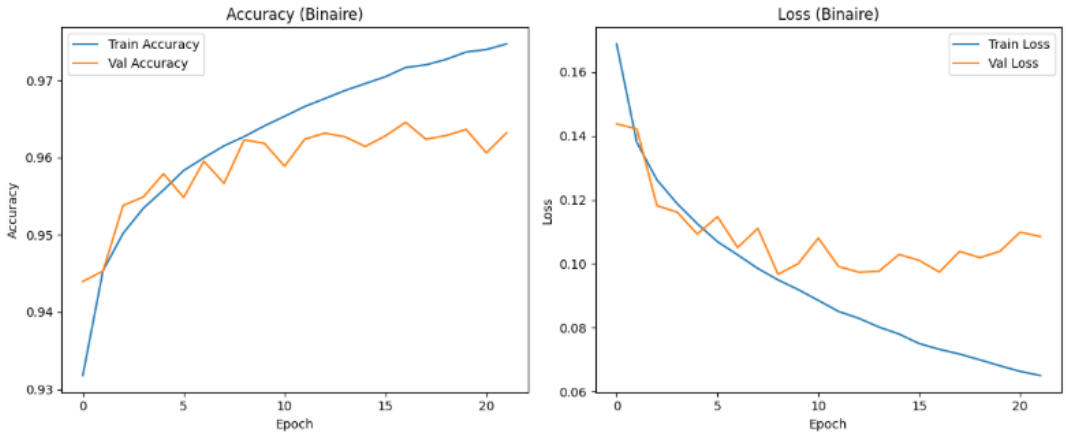
MobileNetV2 achieves the best balance between precision and recall. Its architecture is efficient and generalizes better than deeper models like ResNet, with slightly higher F1-score and more consistent behaviour.

3.4.4. Proposed Custom CNN

Based on the analysis of the above architectures, we designed a lightweight CNN tailored specifically to the binarization problem. It consists of two convolutional blocks followed by dense layers.

a) Training Behaviour

Figure 4.4 illustrates the training behaviour of our custom model:



(a) *(b)*

Figure 4.4. Custom Model Training Progress: (a) training and validation accuracy, (b) training and validation loss

Figure 4.4.a shows that training accuracy rises steadily, reaching 97%, while validation accuracy stabilizes around 96%, indicating excellent generalization and minimal overfitting.

Figure 4.4.b shows consistent decrease in both losses. Training loss reaches 0.06, and validation loss remains between 0.10 and 0.12, confirming stable performance and effective learning.

b) Confusion Matrix and Performance Metrics

Table 4.10. Confusion matrix for Proposed CNN

Actual / Predicted	Background	Foreground	Total
Background	92,185	3,007	95,192
Foreground	2,045	45,310	47,355
Total	94,230	48,317	142,547

Table 4.11. Classification performance for Proposed CNN

Metric	Background	Foreground	Average
Precision	0.9783	0.9378	0.9580
Recall	0.9684	0.9568	0.9626
F1-Score	0.9733	0.9472	**0.9603**

The custom model achieves the best overall performance, surpassing pretrained models in F1-score, precision, and recall for both classes. Its simpler architecture reduces overfitting while maintaining accuracy, making it ideal for practical deployment.

3.4.5. Summary of Results

The comparison of the four models reveals that the proposed CNN achieves the best trade-off between accuracy and computational cost. While MobileNetV2 also performs well, our model's tailored architecture leads to higher F1-scores and more robust generalization, especially for degraded regions.

Thus, the proposed model was selected as the final architecture for our document binarization system.

3.5. Experiment 2: Evaluation of the Proposed System on Full Document Images

After identifying the most effective convolutional neural network (CNN) architecture in Experiment 1, this second experiment aims to evaluate the practical application of the proposed system in binarizing entire historical document images. This phase moves from controlled patch-level classification to full-document processing, providing insights into the model's behavior in realistic conditions.

The evaluation is structured into two complementary experiments:

- **Unblind Testing**, where the model is evaluated on document images from datasets that were partially used during training and validation (specifically DIBCO 2010), and
- **Blind Testing**, where the model is tested on unseen data (H-DIBCO 2016), allowing an unbiased assessment of generalization capabilities.

To ensure a thorough and fair analysis, we also present the results obtained by the other pretrained models tested in Experiment 1—namely, the modernized LeNet, MobileNet, and ResNet architectures. Additionally, we include a comparison with well-known binarization methods from the literature, such as Otsu, Sauvola, and a fully convolutional network (FCN)-based model [44], to situate our approach within the broader context of the state of the art.

3.5.1. Blind Testing: H-DIBCO 2016 Results

The results of the blind evaluation on the H-DIBCO 2016 dataset are summarized in Table 4.12.

Table 4.12. Evaluation metrics for H-DIBCO 2016

Model	F-measure	Pseudo-FM	PSNR	NRM	MPM	DRD
Proposed model	**0.8968**	**0.9385**	**18.7133**	**0.0516**	**0.0049**	**5.1380**
Modernized LeNet	0.8877	0.9192	18.0595	0.0512	0.0048	6.6177
MobileNet	0.8732	0.9141	17.7349	0.0607	0.0052	7.5822
Resnet	0.8667	0.9057	17.2557	0.0605	0.0056	8.6535
Otsu	0.8361	0.8567	16.80	-	-	-
Sauvola	0.8252	0.5685	16.42	-	-	-
FCN-Based Model	0.8952	0.9376	18.67	-	-	3.76

The proposed model demonstrated the best overall performance among the CNN-based architectures tested, surpassing LeNet, ResNet, and MobileNet in nearly all metrics. It achieved a F-measure of 0.8968, indicating a high degree of accuracy in distinguishing foreground text from background regions. The Pseudo-F-measure of 0.9385 also confirms its ability to preserve the readability and structure of text content. The DRD score of 5.1380, although slightly higher than that of the FCN-based model (3.76), remains significantly lower than all other tested models and confirms the visual quality of the binarized output.

3.5.2. Unblind Testing: DIBCO 2010 Results

For additional insight, we evaluated the models on document images from the DIBCO 2010 dataset—images that share characteristics with training data but were not used in testing during training.

Table 4.13. Evaluation metrics for unblind test

Model	F-measure	Pseudo-FM	PSNR	NRM	MPM	DRD
Proposed model	**0.9305**	**0.9511**	**19.3038**	**0.0234**	**0.0010**	**2.7115**
Modernized model	0.8974	0.9048	18.5943	0.0384	0.0020	6.6956
MobileNet20	0.9148	0.9261	19.1072	0.0336	0.0019	5.6276
Resnet20	0.9004	0.9188	18.3403	0.0436	0.0018	6.9459
Otsu	63.47		15.62			
Sauvola	40.36		14.07			
FCN-Based	0.9289	0.9465	19.84			2.86

The results confirm the strong performance of the proposed CNN model in a realistic testing scenario. It achieved the highest F-measure (0.9305) and Pseudo-F-measure (0.9511), confirming its superior capability to distinguish text from background while preserving structural detail. The NRM (0.0234) and MPM (0.0010) scores were the lowest among all models, indicating minimal misclassification and pixel-level noise.

Notably, the DRD score of 2.7115 is significantly lower than those of the other CNN-based models, and only slightly bettered by the FCN-based model (2.86), which benefits from a more

complex architecture. Despite this, the proposed model matches or exceeds the FCN model in most metrics while remaining computationally lightweight.

Compared to traditional binarization methods such as Otsu and Sauvola, which scored only 0.6347 and 0.4036 in F-measure, respectively, the CNN-based methods provide a substantial improvement in binarization quality. These classical methods failed to handle degradation and background variation effectively, resulting in poor performance.

In conclusion, these unblind test results demonstrate that the proposed system not only generalizes well to data similar to the training distribution but also offers a competitive balance between performance and efficiency. Its consistent outperformance of both pretrained models and traditional methods makes it a strong candidate for real-world historical document binarization tasks.

4. Application

The developed application is composed of three key interfaces, each corresponding to a major phase of the proposed system: data preparation, model training, and document image binarization. Together, they provide an end-to-end framework that facilitates experimentation, model development, and evaluation in a user-friendly manner.

4.1. Data Preparation Interface

This interface is dedicated to the preparation of training data. It enables users to extract representative 32×32 patches from historical document images, a critical step for effective pixel-level classification. The process begins with dimensionality reduction using *Principal Component Analysis (PCA)*, followed by clustering using the *KMeans* algorithm to group similar foreground and background patches. As a result, the extracted samples are organized into two distinct folders—foreground and background—which can later be used as input for CNN training.

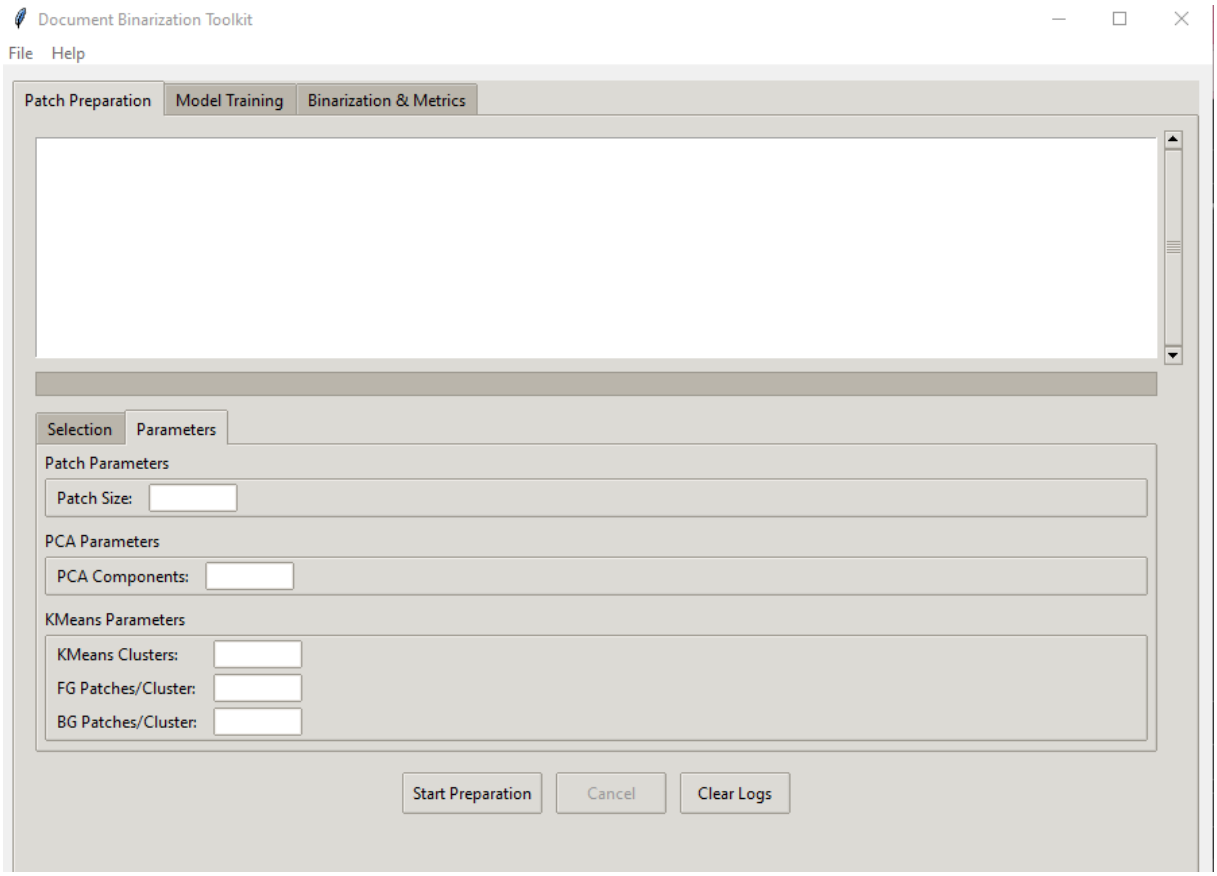


Figure 4.5. Preparation interface

4.2. Model Training Interface

This interface facilitates the training of convolutional neural network (CNN) models using the prepared patch dataset. Users can specify training parameters, such as the number of epochs and whether to enable data augmentation. The interface offers real-time visual feedback by displaying training and validation accuracy and loss curves, thereby enabling users to monitor learning progress and detect issues such as overfitting or underfitting. This interactive environment simplifies experimentation with different model configurations.

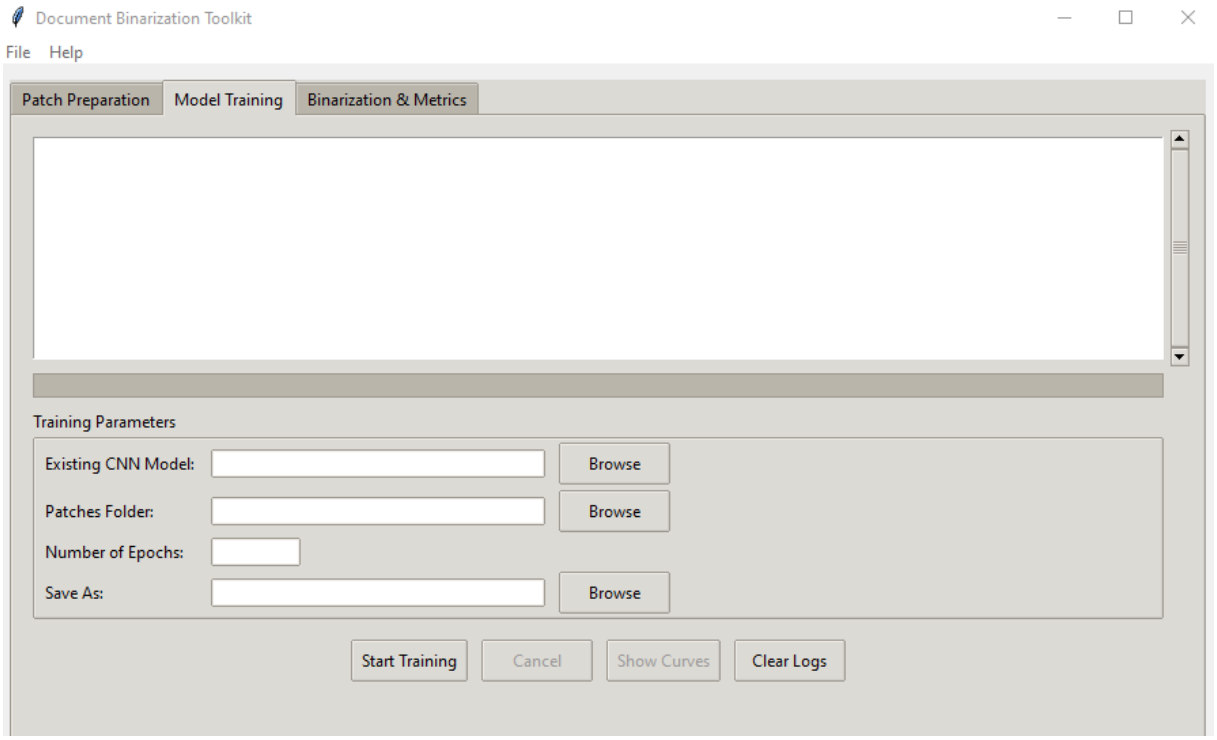


Figure 4.6. Training interface

4.3. Binarization Interface

The final interface supports the practical application of trained models for document image binarization. Users can select a new document image and apply the trained model in a sliding-window fashion to generate a binarized output. If the ground truth for the document is available, the system also computes evaluation metrics such as F-measure, PSNR, and DRD. To facilitate visual assessment, the interface displays the original image, the binarized output, and the ground truth image side-by-side. This allows users to qualitatively and quantitatively evaluate the binarization quality.

This interface provides tools for applying trained models to perform binarization of new document images. When the ground truth is available, it also computes evaluation metrics. The interface offers a visual comparison by displaying the original input, the predicted binarized output, and the corresponding ground truth image side-by-side, making it easier to assess model performance.

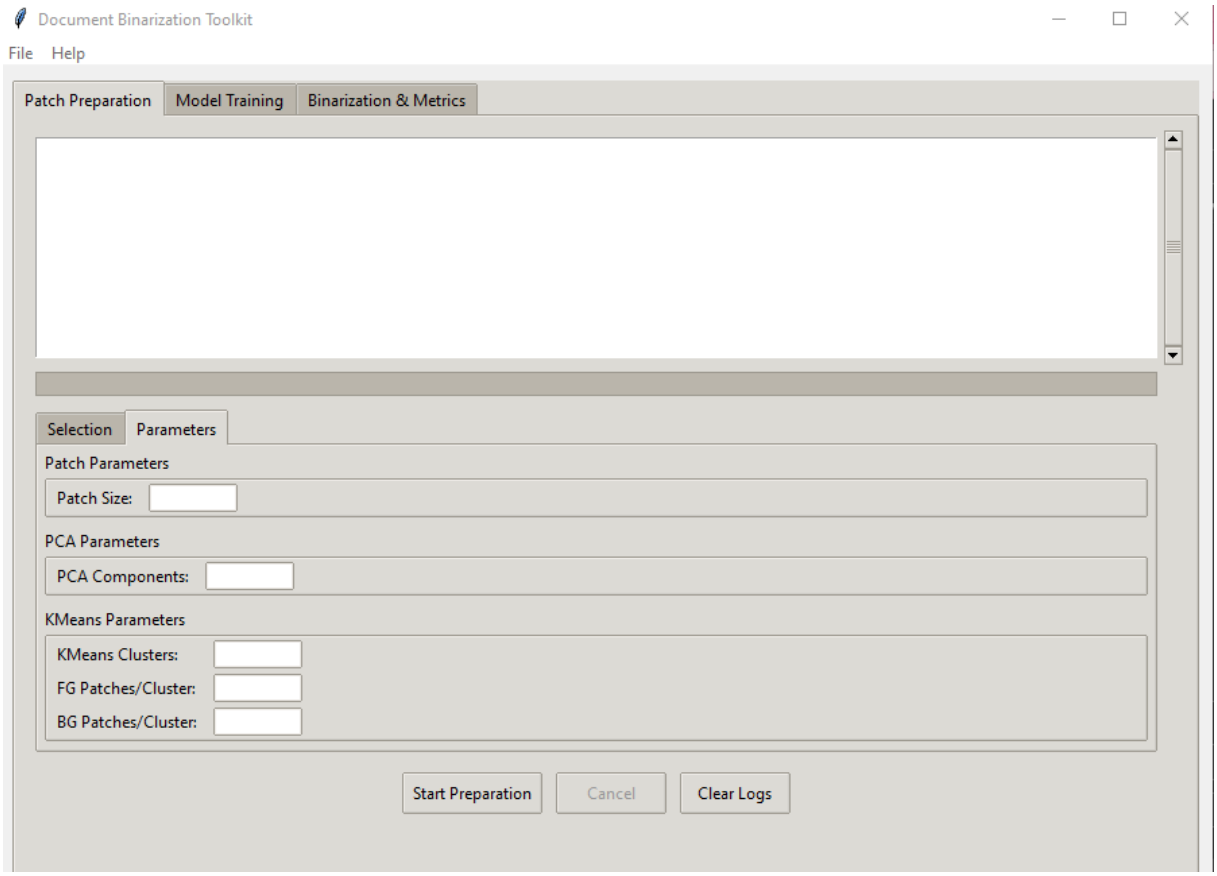


Figure 4.7. Evaluation and binarization interface

5. Conclusion

Throughout this project, we developed and tested multiple deep learning models tailored for the binarization of historical document images. The implementation involved experimenting with various architectures, optimizing training strategies, and carefully evaluating each model's performance using relevant quality metrics. Among the tested models, some demonstrated better generalization and stability, proving more effective in handling the complexity of degraded documents.

In addition to the models, a fully functional application was created to streamline the processing pipeline. This tool allows users to prepare data, load models, perform inference, and visualize results interactively. Overall, both the models and the application contribute to an efficient and practical approach to document image binarization, confirming the feasibility and effectiveness of deep learning in this domain.

General Conclusion and Perspectives

Conclusion

This dissertation has addressed the problem of binarizing degraded historical document images using a classification-based approach grounded in convolutional neural networks (CNNs). The proposed methodology was designed to overcome the limitations of traditional binarization techniques, particularly in handling various types of degradation, such as ink bleed-through, uneven illumination, and background noise.

We began by formulating the binarization task from a pixel-level classification perspective, where each pixel is assigned a label—foreground (text) or background (non-text)—based on its visual and contextual characteristics. To support this strategy, we constructed a carefully curated dataset composed of representative image patches extracted from annotated historical documents. Dimensionality reduction (PCA) and clustering (K-Means) were applied to ensure diversity and compactness in the training data.

Next, we explored different CNN-based models for this task. After experimenting with several pretrained architectures, including a modernized LeNet, a customized ResNet, and a modified MobileNet, we proposed a lightweight custom CNN tailored specifically for the 32×32 grayscale patches used in this work. The trained classifier was then used to produce binarized versions of full document images by classifying patches in a sliding-window manner.

A series of experiments were conducted to rigorously evaluate and optimize the system. These included testing various pretrained models, progressively refining the custom architecture, and adjusting critical training parameters to select the optimal configuration. Furthermore, extensive evaluations were performed both on unseen (blind) and known (non-blind) data to comprehensively assess the generalization ability and robustness of the proposed approach.

The results demonstrated that our custom model achieved strong performance across several evaluation metrics, including F1-score, PSNR, NRM, MPM, and DRD, outperforming the compared methods. It showed a high capacity for accurately distinguishing between degraded text and background regions, while also maintaining computational efficiency. When applied to blind datasets—consisting of entirely unseen historical document pages—the model consistently delivered high classification accuracy, underscoring its robustness and generalizability. The resulting binarized images were visually clean, with fine textual details preserved and background noise effectively suppressed.

These encouraging results confirm the effectiveness of or methodology in real-world settings and its relevance to practical applications in document analysis, digitization, and OCR preprocessing.

Perspectives

While the results obtained are promising, this work also opens up several research directions and improvements:

1. **Incorporation of Spatial Context Beyond Patch Boundaries**: The current approach classifies each patch independently. Extending the model to incorporate broader spatial dependencies—possibly through fully convolutional networks (FCNs) or transformer-based models—could further improve segmentation quality.

2. **Integration with Document Analysis Pipelines**: The binarization system could be embedded into larger pipelines for layout analysis, optical character recognition (OCR), or document restoration, enhancing end-to-end performance in real-world applications.
3. **Data Augmentation and Synthetic Degradations**: Generating additional training samples using data augmentation or synthetic degradation techniques could help the model become more robust to rare or extreme degradation patterns.
4. **Cross-Dataset Generalization**: Further evaluation on different historical document datasets would help assess the generality of the model and potentially enable the development of a universal binarization solution.
5. **Real-Time and Lightweight Deployment**: Optimizing the model for inference speed and memory usage could enable deployment on embedded systems, mobile devices, or scanners for real-time document digitization.

References

References

[1] P. (0427?-0348? av J.-C.) A. du texte, La république (livre VI) (Nouvelle édition, avec une étude sur la politique platonicienne, une analyse sommaire du dialogue “la République”, des notes historiques et philosophiques et un extrait du livre VII) / Platon, 1886. https://gallica.bnf.fr/ark:/12148/bpt6k1194108s (accessed June 28, 2025).

[2] B. Benchamardimath, R. Hegadi, A STUDY ON THE IMPORTANCE OF IMAGE PROCESSING AND ITS APLLICATIONS, IJRET: International Journal of Research in Engineering and Technology 3 (2014).

[3] S.D. Johnson, P.-A. Moreau, T. Gregory, M.J. Padgett, How many photons does it take to form an image?, Appl. Phys. Lett. 116 (2020). https://doi.org/10.1063/5.0009493.

[4] M. Obulesu, V. Kishore, A New Approach for Sharpness and Contrast Enhancement of an Image, International Journal of Advanced Research in Computer Engineering & Technology 1 (2012) 530–535.

[5] K. Tran, J.P. Bøtker, A. Aframian, K. Memarzadeh, Chapter 6 - Artificial intelligence for medical imaging, in: A. Bohr, K. Memarzadeh (Eds.), Artificial Intelligence in Healthcare, Academic Press, 2020: pp. 143–162. https://doi.org/10.1016/B978-0-12-818438-7.00006-X.

[6] R.S. Dwivedi, Digital Image Processing, in: D. Ravi Shankar (Ed.), Remote Sensing of Soils, Springer, Berlin, Heidelberg, 2017: pp. 117–165. https://doi.org/10.1007/978-3-662-53740-4_3.

[7] S. Roy, A. Mitra, S.K. Setua, Color Image Representation Using Multivector, in: Modelling and Simulation 2014 5th International Conference on Intelligent Systems, 2014: pp. 357–363. https://doi.org/10.1109/ISMS.2014.66.

[8] L. O’GORMAN, R. KASTURI, Document image analysis, Los Alamitos: IEEE Computer Society Press, 1995.

[9] K. Sun, G. Cao, Q. Zhao, J. Zhang, Differential Abnormality-Based Tampering Detection in Digital Document Images, in: 2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS), 2019: pp. 145–149. https://doi.org/10.1109/ICIS46139.2019.8940190.

[10] P.K. LOO, DOCUMENT IMAGE PROCESSING USING IRREGULAR PYRAMID STRUCTURE, NATIONAL UNIVERSITY OF SINGAPORE, 2004.

[11] S. Marinai, Chapter 16 - Learning Algorithms for Document Layout Analysis, in: C.R. Rao, V. Govindaraju (Eds.), Handbook of Statistics, Elsevier, 2013: pp. 400–419. https://doi.org/10.1016/B978-0-444-53859-8.00016-3.

[12] A. Sulaiman, K. Omar, M.F. Nasrudin, Degraded Historical Document Binarization: A Review on Issues, Challenges, Techniques, and Future Directions, Journal of Imaging 5 (2019) 48. https://doi.org/10.3390/jimaging5040048.

[13] J. Flusser, M. Lébl, F. Šroubek, M. Pedone, J. Kostková, Blur Invariants for Image Recognition, Int J Comput Vis 131 (2023) 2298–2315. https://doi.org/10.1007/s11263-023-01798-7.

[14] C. Tensmeyer, T. Martinez, Historical Document Image Binarization: A Review, SN COMPUT. SCI. 1 (2020) 173. https://doi.org/10.1007/s42979-020-00176-1.

References

[15] A. Sehad, Ancient degraded document image binarization based on texture features, 2013.

[16] K. Khurshid, I. Siddiqi, C. Faure, N. Vincent, Comparison of Niblack inspired binarization methods for ancient documents, in: Document Recognition and Retrieval XVI, SPIE, 2009: pp. 267–275. https://doi.org/10.1117/12.805827.

[17] O.D. Trier, T. Taxt, Evaluation of binarization methods for document images, IEEE Transactions on Pattern Analysis and Machine Intelligence 17 (1995) 312–315. https://doi.org/10.1109/34.368197.

[18] D. Kumar, M.N.A. Prasad, A.G. Ramakrishnan, Evaluation of document binarization using eigen value decomposition, in: Document Recognition and Retrieval XX, SPIE, 2013: pp. 308–319. https://doi.org/10.1117/12.2008502.

[19] B. Fernando, S. Karaoglu, A. Trémeau, Extreme value theory based text binarization in documents and natural scenes, in: 2010: p. 144. https://hal.science/hal-00590644 (accessed January 28, 2025).

[20] M. Sezgin, B. Sankur, Survey over image thresholding techniques and quantitative performance evaluation, JEI 13 (2004) 146–165. https://doi.org/10.1117/1.1631315.

[21] Z. Yang, S. Zuo, Y. Zhou, J. He, J. Shi, A Review of Document Binarization: Main Techniques, New Challenges, and Trends, Electronics 13 (2024) 1394. https://doi.org/10.3390/electronics13071394.

[22] M.R. Gupta, N.P. Jacobson, E.K. Garcia, OCR binarization and image pre-processing for searching historical documents, Pattern Recognition 40 (2007) 389–397. https://doi.org/10.1016/j.patcog.2006.04.043.

[23] N. Otsu, A Tlreshold Selection Method from Gray-Level Histograms, (n.d.).

[24] J.N. Kapur, P.K. Sahoo, A.K.C. Wong, A new method for gray-level picture thresholding using the entropy of the histogram, Computer Vision, Graphics, and Image Processing 29 (1985) 273–285. https://doi.org/10.1016/0734-189X(85)90125-2.

[25] F.R.D. Velasco, Thresholding using the ISODATA clustering algorithm, IEEE Transactions on Systems Man and Cybernetics 10 (1980) 771–774.

[26] W. Niblack, An introduction to digital image processing, Strandberg Publishing Company, DNK, 1985.

[27] J. Sauvola, T. Seppanen, S. Haapakoski, M. Pietikainen, Adaptive document binarization, in: Proceedings of the Fourth International Conference on Document Analysis and Recognition, 1997: pp. 147–152 vol.1. https://doi.org/10.1109/ICDAR.1997.619831.

[28] A. Kefali, T. Sari, Et, M. Sellami, Evaluation de plusieurs techniques de seuillage d'images de documents Arabes anciens, 2009.

[29] B. Bataineh, S. Abdullah, K. Omar, M.F. Nasrudin, Adaptive Thresholding Methods for Documents Image Binarization, 2011. https://doi.org/10.1007/978-3-642-21587-2_25.

[30] S. N, V. S, Image Segmentation By Using Thresholding Techniques For Medical Images, CSEIJ 6 (2016) 1–13. https://doi.org/10.5121/cseij.2016.6101.

References

[31] K. Saddami, F. Arnia, Y. Away, K. Munadi, Kombinasi Metode Nilai Ambang Lokal dan Global untuk Restorasi Dokumen Jawi Kuno, Jurnal Teknologi Informasi dan Ilmu Komputer 7 (2020) 163–170. https://doi.org/10.25126/jtiik.2020701741.

[32] E. Zemouri, Enhancement of Historical Document Images by Combining Global and Local Binarization Technique, IJIEE 4 (2014). https://doi.org/10.7763/IJIEE.2014.V4.397.

[33] K. Ntirogiannis, B. Gatos, I. Pratikakis, A combined approach for the binarization of handwritten document images, Pattern Recognition Letters 35 (2014) 3–15. https://doi.org/10.1016/j.patrec.2012.09.026.

[34] K. Kita, T. Wakahara, Binarization of Color Characters in Scene Images Using k-means Clustering and Support Vector Machines, in: 2010 20th International Conference on Pattern Recognition, IEEE, Istanbul, Turkey, 2010: pp. 3183–3186. https://doi.org/10.1109/ICPR.2010.779.

[35] C.-H. Chou, W.-H. Lin, F. Chang, A binarization method with learning-built rules for document images produced by cameras, Pattern Recognition 43 (2010) 1518–1530. https://doi.org/10.1016/j.patcog.2009.10.016.

[36] T. Chen, M. Takagi, Image Binarization by Back Propagation Algorithm, in: 1993: pp. 345–345.

[37] A. Khashman, B. Sekeroglu, Global Binarization of Document Images Using a Neural Network, in: 2007 Third International IEEE Conference on Signal-Image Technologies and Internet-Based System, 2007: pp. 665–672. https://doi.org/10.1109/SITIS.2007.58.

[38] J. Lázaro, J.L. Martín, J. Arias, A. Astarloa, C. Cuadrado, Neuro semantic thresholding using OCR software for high precision OCR applications, Image and Vision Computing 28 (2010) 571–578. https://doi.org/10.1016/j.imavis.2009.09.011.

[39] A. Kefali, T. Sari, H. Bahi, Foreground-Background Separation by Feed-forward Neural Networks in Old Manuscripts, Informatica 38 (2014). https://informatica.si/index.php/informatica/article/view/715 (accessed February 26, 2025).

[40] J. Pastor-Pellicer, S. España-Boquera, F. Zamora-Martínez, M.Z. Afzal, M.J. Castro-Bleda, Insights on the Use of Convolutional Neural Networks for Document Image Binarization, 2015. https://doi.org/10.1007/978-3-319-19222-2_10.

[41] Q.N. Vo, S.H. Kim, H.J. Yang, G. Lee, Binarization of degraded document images based on hierarchical deep supervised network, Pattern Recognition 74 (2018) 568–586. https://doi.org/10.1016/j.patcog.2017.08.025.

[42] S. He, L. Schomaker, DeepOtsu: Document enhancement and binarization using iterative deep learning, Pattern Recognition 91 (2019) 379–390. https://doi.org/10.1016/j.patcog.2019.01.025.

[43] J. Long, E. Shelhamer, T. Darrell, Fully Convolutional Networks for Semantic Segmentation, (n.d.).

References

[44] C. Tensmeyer, T. Martinez, Document Image Binarization with Fully Convolutional Neural Networks, in: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), 2017: pp. 99–104. https://doi.org/10.1109/ICDAR.2017.25.

[45] J. Calvo-Zaragoza, A.-J. Gallego, A selectional auto-encoder approach for document image binarization, Pattern Recognition 86 (2019) 37–47. https://doi.org/10.1016/j.patcog.2018.08.011.

[46] S. Suh, J. Kim, P. Lukowicz, Y.O. Lee, Two-stage generative adversarial networks for document image binarization with color noise and background removal, (2021). https://doi.org/10.48550/arXiv.2010.10103.

[47] A.K. Bhunia, A.K. Bhunia, A. Sain, P.P. Roy, Improving Document Binarization Via Adversarial Noise-Texture Augmentation, in: 2019 IEEE International Conference on Image Processing (ICIP), 2019: pp. 2721–2725. https://doi.org/10.1109/ICIP.2019.8803348.

[48] R. De, A. Chakraborty, R. Sarkar, Document Image Binarization Using Dual Discriminator Generative Adversarial Networks, IEEE Signal Processing Letters 27 (2020) 1090–1094. https://doi.org/10.1109/LSP.2020.3003828.

[49] F. Westphal, N. Lavesson, H. Grahn, Document Image Binarization Using Recurrent Neural Networks, in: 2018 13th IAPR International Workshop on Document Analysis Systems (DAS), IEEE, Vienna, 2018: pp. 263–268. https://doi.org/10.1109/DAS.2018.71.

[50] B. Gatos, K. Ntirogiannis, I. Pratikakis, ICDAR 2009 Document Image Binarization Contest (DIBCO 2009), in: 2009 10th International Conference on Document Analysis and Recognition, 2009: pp. 1375–1382. https://doi.org/10.1109/ICDAR.2009.246.

[51] I. Pratikakis, B. Gatos, K. Ntirogiannis, H-DIBCO 2010 - Handwritten Document Image Binarization Competition, in: 2010 12th International Conference on Frontiers in Handwriting Recognition, 2010: pp. 727–732. https://doi.org/10.1109/ICFHR.2010.118.

[52] I. Pratikakis, B. Gatos, K. Ntirogiannis, ICDAR 2011 Document Image Binarization Contest (DIBCO 2011), in: 2011 International Conference on Document Analysis and Recognition, 2011: pp. 1506–1510. https://doi.org/10.1109/ICDAR.2011.299.

[53] I. Pratikakis, B. Gatos, K. Ntirogiannis, ICFHR 2012 Competition on Handwritten Document Image Binarization (H-DIBCO 2012), in: 2012 International Conference on Frontiers in Handwriting Recognition, 2012: pp. 817–822. https://doi.org/10.1109/ICFHR.2012.216.

[54] I. Pratikakis, B. Gatos, K. Ntirogiannis, ICDAR 2013 Document Image Binarization Contest (DIBCO 2013), in: 2013 12th International Conference on Document Analysis and Recognition, 2013: pp. 1471–1476. https://doi.org/10.1109/ICDAR.2013.219.

[55] K. Ntirogiannis, B. Gatos, I. Pratikakis, ICFHR2014 Competition on Handwritten Document Image Binarization (H-DIBCO 2014), in: 2014 14th International Conference on Frontiers in Handwriting Recognition, 2014: pp. 809–813. https://doi.org/10.1109/ICFHR.2014.141.

[56] I. Pratikakis, K. Zagoris, G. Barlas, B. Gatos, ICFHR2016 Handwritten Document Image Binarization Contest (H-DIBCO 2016), in: 2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR), 2016: pp. 619–623. https://doi.org/10.1109/ICFHR.2016.0118.

[57] I. Pratikakis, K. Zagoris, G. Barlas, B. Gatos, ICDAR2017 Competition on Document Image Binarization (DIBCO 2017), in: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), 2017: pp. 1395–1403. https://doi.org/10.1109/ICDAR.2017.228.

[58] I. Pratikakis, K. Zagori, P. Kaddas, B. Gatos, ICFHR 2018 Competition on Handwritten Document Image Binarization (H-DIBCO 2018), in: 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), 2018: pp. 489–493. https://doi.org/10.1109/ICFHR-2018.2018.00091.

[59] I. Pratikakis, K. Zagoris, X. Karagiannis, L. Tsochatzidis, T. Mondal, I. Marthot-Santaniello, ICDAR 2019 Competition on Document Image Binarization (DIBCO 2019), in: 2019 International Conference on Document Analysis and Recognition (ICDAR), 2019: pp. 1547–1556. https://doi.org/10.1109/ICDAR.2019.00249.

[60] S.M. Ayatollahi, H. Ziaei Nafchi, Persian heritage image binarization competition (PHIBC 2012), in: 2013 First Iranian Conference on Pattern Recognition and Image Analysis (PRIA), 2013: pp. 1–4. https://doi.org/10.1109/PRIA.2013.6528442.

[61] A.F. Mollah, S. Basu, M. Nasipuri, Computationally Efficient Implementation of Convolution-Based Locally Adaptive Binarization Techniques, in: K.R. Venugopal, L.M. Patnaik (Eds.), Wireless Networks and Computational Intelligence, Springer, Berlin, Heidelberg, 2012: pp. 159–168. https://doi.org/10.1007/978-3-642-31686-9_18.

[62] B. Su, S. Lu, C.L. Tan, Robust Document Image Binarization Technique for Degraded Document Images, IEEE Transactions on Image Processing 22 (2013) 1408–1417. https://doi.org/10.1109/TIP.2012.2231089.

[63] R. Rowley-Brooke, F. Pitié, A. Kokaram, A Ground Truth Bleed-Through Document Image Database, in: P. Zaphiris, G. Buchanan, E. Rasmussen, F. Loizides (Eds.), Theory and Practice of Digital Libraries, Springer, Berlin, Heidelberg, 2012: pp. 185–196. https://doi.org/10.1007/978-3-642-33290-6_21.

[64] P. Ye, D. Doermann, Document Image Quality Assessment: A Brief Survey, in: 2013 12th International Conference on Document Analysis and Recognition, 2013: pp. 723–727. https://doi.org/10.1109/ICDAR.2013.148.

[65] Y. ZHANG, New Advances in Machine Learning, BoD–Books on Demand, 2010.

[66] B.K. Pandey, D. Pandey, R. Anand, D.S. Mane, V.K. Nassa, Handbook of Research on Thrust Technologies' Effect on Image Processing, IGI Global Scientific Publishing, 1AD. https://www.igi-global.com/book/handbook-research-thrust-technologies-effect/www.igi-global.com/book/handbook-research-thrust-technologies-effect/314629 (accessed June 29, 2025).

[67] V. Nasteski, An overview of the supervised machine learning methods, HORIZONS.B 4 (2017) 51–62. https://doi.org/10.20544/HORIZONS.B.04.1.17.P05.

[68] What is Unsupervised Learning?, GeeksforGeeks (19:31:27+00:00). https://www.geeksforgeeks.org/unsupervised-learning/ (accessed May 27, 2025).

[69] M. Usama, J. Qadir, A. Raza, H. Arif, K.A. Yau, Y. Elkhatib, A. Hussain, A. Al-Fuqaha, Unsupervised Machine Learning for Networking: Techniques, Applications and Research Challenges, IEEE Access 7 (2019) 65579–65615. https://doi.org/10.1109/ACCESS.2019.2916648.

[70] Y. Li, Deep Reinforcement Learning: An Overview, (2018). https://doi.org/10.48550/arXiv.1701.07274.

[71] A. Mathew, P. Amudha, S. Sivakumari, Deep Learning Techniques: An Overview, in: A.E. Hassanien, R. Bhatnagar, A. Darwish (Eds.), Advanced Machine Learning Technologies and Applications, Springer, Singapore, 2021: pp. 599–608. https://doi.org/10.1007/978-981-15-3383-9_54.

[72] M. Coşkun, Ö. Yıldırım, A. Uçar, Y. Demır, AN OVERVIEW OF POPULAR DEEP LEARNING METHODS, EJT 7 (2017) 165–176.

[73] S. Dargan, M. Kumar, M.R. Ayyagari, G. Kumar, A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning, Arch Computat Methods Eng 27 (2020) 1071–1092. https://doi.org/10.1007/s11831-019-09344-w.

[74] S. Shetty, A. Siddiqa, Deep Learning Algorithms and Applications in Computer Vision, International Journal of Computer Sciences and Engineering 7 (2019) 195–201. https://doi.org/10.26438/ijcse/v7i7.195201.

[75] A. Upreti, Convolutional Neural Network (CNN). A Comprehensive Overview, (2022). https://doi.org/10.20944/preprints202208.0313.v3.

[76] J. Ayeni, Convolutional Neural Network (CNN): The architecture and applications, Applied Journal of Physical Science 4 (2022) 42–50. https://doi.org/10.31248/AJPS2022.085.

[77] A. Alsaleh, C. Perkgoz, A space and time efficient convolutional neural network for age group estimation from facial images, PeerJ Comput. Sci. 9 (2023) e1395. https://doi.org/10.7717/peerj-cs.1395.

[78] N. Aloysius, M. Geetha, A review on deep convolutional neural networks, in: 2017 International Conference on Communication and Signal Processing (ICCSP), 2017: pp. 0588–0592. https://doi.org/10.1109/ICCSP.2017.8286426.

[79] L. Alzubaidi, J. Zhang, A.J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M.A. Fadhel, M. Al-Amidie, L. Farhan, Review of deep learning: concepts, CNN architectures, challenges, applications, future directions, J Big Data 8 (2021) 53. https://doi.org/10.1186/s40537-021-00444-8.

[80] O. Elharrouss, Y. Mahmood, Y. Bechqito, M.A. Serhani, E. Badidi, J. Riffi, H. Tairi, Loss Functions in Deep Learning: A Comprehensive Review, (2025). https://doi.org/10.48550/arXiv.2504.04242.

[81] A. Jadon, A. Patil, S. Jadon, A Comprehensive Survey of Regression Based Loss Functions for Time Series Forecasting, (2022). https://doi.org/10.48550/arXiv.2211.02989.

[82] Binary Cross Entropy/Log Loss for Binary Classification, GeeksforGeeks (15:11:55+00:00). https://www.geeksforgeeks.org/binary-cross-entropy-log-loss-for-binary-classification/ (accessed June 13, 2025).

[83] O. Elharrouss, Y. Mahmood, Y. Bechqito, M.A. Serhani, E. Badidi, J. Riffi, H. Tairi, Loss Functions in Deep Learning: A Comprehensive Review, (2025). https://doi.org/10.48550/arXiv.2504.04242.

[84] N. Gowdra, R. Sinha, S. MacDonell, W. Yan, Maximum Categorical Cross Entropy (MCCE): A noise-robust alternative loss function to mitigate racial bias in Convolutional Neural Networks (CNNs) by reducing overfitting, (2020). https://openreview.net/forum?id=1IBgFQbj7y (accessed June 13, 2025).

[85] D. Soydaner, A Comparison of Optimization Algorithms for Deep Learning, Int. J. Patt. Recogn. Artif. Intell. 34 (2020) 2052013. https://doi.org/10.1142/S0218001420520138.

[86] M. Swapna, Dr.Y. Sharma, B. Prasad, CNN Architectures: Alex Net, Le Net, VGG, Google Net, Res Net, International Journal of Recent Technology and Engineering (IJRTE) 8 (2020) 953–959. https://doi.org/10.35940/ijrte.F9532.038620.

[87] M. Rybczak, K. Kozakiewicz, Deep Machine Learning of MobileNet, Efficient, and Inception Models, Algorithms 17 (2024) 96. https://doi.org/10.3390/a17030096.

[88] S. Salman, X. Liu, Overfitting Mechanism and Avoidance in Deep Neural Networks, (2019). https://doi.org/10.48550/arXiv.1901.06566.

[89] L. Prechelt, Early Stopping - But When?, in: G.B. Orr, K.-R. Müller (Eds.), Neural Networks: Tricks of the Trade, Springer, Berlin, Heidelberg, 1998: pp. 55–69. https://doi.org/10.1007/3-540-49430-8_3.

[90] S.F.G. Dos, P. Paulo, Avoiding Overfitting: A Survey on Regularization Methods for Convolutional Neural Networks, ACM Computing Surveys (CSUR) (2022). https://doi.org/10.1145/3510413.

[91] Z. abu bakar, M. Mat Deris, A. Che Alhadi, Performance Analysis of Partitional and Incremental Clustering, 0 (2005).

[92] S. Pitafi, T. Anwar, Z. Sharif, A Taxonomy of Machine Learning Clustering Algorithms, Challenges, and Future Realms, Applied Sciences 13 (2023) 3529. https://doi.org/10.3390/app13063529.

[93] E.U. Oti, M.O. Olusola, F.C. Eze, S.U. Enogwe, Comprehensive Review of K-Means Clustering Algorithms, IJASRE 07 (2021) 64–69. https://doi.org/10.31695/IJASRE.2021.34050.

[94] M.E. Celebi, H.A. Kingravi, P.A. Vela, A comparative study of efficient initialization methods for the k-means clustering algorithm, Expert Systems with Applications 40 (2013) 200–210. https://doi.org/10.1016/j.eswa.2012.07.021.

[95] T. Kanungo, D.M. Mount, N.S. Netanyahu, C. Piatko, R. Silverman, A.Y. Wu, The analysis of a simple *k* -means clustering algorithm, in: Proceedings of the Sixteenth Annual Symposium on Computational Geometry, ACM, Clear Water Bay Kowloon Hong Kong, 2000: pp. 100–109. https://doi.org/10.1145/336154.336189.

References

[96] R. Zebari, A. Abdulazeez, D. Zeebaree, D. Zebari, J. Saeed, A Comprehensive Review of Dimensionality Reduction Techniques for Feature Selection and Feature Extraction, Journal of Applied Science and Technology Trends 1 (2020) 56–70. https://doi.org/10.38094/jastt1224.

[97] B.M.S. Hasan, A.M. Abdulazeez, A Review of Principal Component Analysis Algorithm for Dimensionality Reduction, Journal of Soft Computing and Data Mining 2 (2021) 20–30.

[98] P.J.M. Ali, R.H. Faraj, P.J.M. Ali, R.H. Faraj, A. Technical, Data Normalization and Standardization: A Technical Report, (n.d.).

[99] G.K. Baydoğmuş, The Effects of Normalization and Standardization an Internet of Things Attack Detection, EJOSAT (2021) 187–192. https://doi.org/10.31590/ejosat.1017427.

[100] P.-O. Côté, A. Nikanjam, N. Ahmed, D. Humeniuk, F. Khomh, Data cleaning and machine learning: a systematic literature review, Autom Softw Eng 31 (2024) 54. https://doi.org/10.1007/s10515-024-00453-w.

[101] M. Afkanpour, E. Hosseinzadeh, H. Tabesh, Identify the most appropriate imputation method for handling missing values in clinical structured datasets: a systematic review, BMC Med Res Methodol 24 (2024) 188. https://doi.org/10.1186/s12874-024-02310-6.

[102] Z. Wang, P. Wang, K. Liu, P. Wang, Y. Fu, C.-T. Lu, C.C. Aggarwal, J. Pei, Y. Zhou, A Comprehensive Survey on Data Augmentation, (2025). https://doi.org/10.48550/arXiv.2405.09591.

[103] Y. Dong, Beating Kaggle the easy way, (n.d.).

[104] Notebooks Documentation, (n.d.). https://www.kaggle.com/docs/notebooks (accessed June 16, 2025).

[105] What is TensorFlow? | Definition from TechTarget, Search Data Management (n.d.). https://www.techtarget.com/searchdatamanagement/definition/TensorFlow (accessed June 16, 2025).

[106] What is NumPy? — NumPy v2.3 Manual, (n.d.). https://numpy.org/doc/stable/user/whatisnumpy.html (accessed June 16, 2025).

[107] About, OpenCV (n.d.). https://opencv.org/about/ (accessed June 16, 2025).

[108] Pillow, Pillow (PIL Fork) (n.d.). https://pillow.readthedocs.io/en/stable/index.html (accessed June 16, 2025).

[109] K. Team, Keras documentation: About Keras 3, (n.d.). https://keras.io/getting_started/about/ (accessed June 16, 2025).

[110] Getting Started, Scikit-Learn (n.d.). https://scikit-learn/stable/getting_started.html (accessed June 16, 2025).

[111] Matplotlib — Visualization with Python, (n.d.). https://matplotlib.org/ (accessed June 16, 2025).

[112] random — Generate pseudo-random numbers, Python Documentation (n.d.). https://docs.python.org/3/library/random.html (accessed June 16, 2025).

[113] doxapy: An image binarization library focussing on local adaptive thresholding, (n.d.). https://github.com/brandonmpetty/doxa (accessed June 16, 2025).

[114] S. Van Der Walt, J.L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J.D. Warner, N. Yager, E. Gouillart, T. Yu, scikit-image: image processing in Python, PeerJ 2 (2014) e453. https://doi.org/10.7717/peerj.453.

[115] R. Dueire Lins, E. Kavallieratou, E.B. Smith, R.B. Bernardino, D.M. de Jesus, ICDAR 2019 Time-Quality Binarization Competition, in: 2019 International Conference on Document Analysis and Recognition (ICDAR), 2019: pp. 1539–1546. https://doi.org/10.1109/ICDAR.2019.00248.

[116] I. Markoulidakis, G. Markoulidakis, Probabilistic Confusion Matrix: A Novel Method for Machine Learning Algorithm Generalized Performance Analysis, Technologies 12 (2024) 113. https://doi.org/10.3390/technologies12070113.

[117] A. Baratloo, M. Hosseini, A. Negida, G. El Ashal, Part 1: Simple Definition and Calculation of Accuracy, Sensitivity and Specificity, Emerg (Tehran) 3 (2015) 48–49.

[118] Jyotsna, S. Chauhan, E. Sharma, A. Doegar, Binarization techniques for degraded document images — A review, in: 2016 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2016: pp. 163–166. https://doi.org/10.1109/ICRITO.2016.7784945.